Imperial College London

A brief introduction to numerical simulation & Fluidity

Matthew Piggott
m.d.piggott@imperial.ac.uk

This presentation

- An overview of Fluidity
- Some very brief comments on things to consider when conducting numerical simulations
- Issues you need to think about
- Things you need to be wary of

- This is a personal view of this area, it does not replace a full course on numerical analysis, numerical methods for (O or P) DEs, etc, where many of these concepts would be covered in more detail

- Remember that many of the concepts related to stability, convergence, order, best practice, stabilisation, etc, are common to many methods / codes

The Name

- Fluidity, ICOM, Fluidity-ICOM?
- For a number of reasons, some good, some not so good, a number of names have been used for the code we're discussing here. They are all, and always have been, one and the same thing!
- If an application was CFD-like we tended to use 'Fluidity'
- If an application was GFD-like we tended to use 'ICOM'
- Now that the code is released to multiple communities we need to clarify things to avoid confusion
- The current convention is to use 'Fluidity' as the name of the CFD model as well as the overall modelling (code) framework
- and to use Fluidity-ICOM for GFD / ocean applications
- and Fluidity-**** for other application areas under development (e.g. **** could take the form: Stokes, Rocks, Solids, Atmos, ICAM, RT, ... in the future, yet to be decided)

What is Fluidity?

- A whole bunch of Fortran, C/C++ and Python*
- linked to a number of external libraries
- to solve a variety of PDE systems
- using control volume / finite element discretisation methods
- on unstructured (adapting) meshes
- in parallel
- coupled to external models / codes as well as internal modules

- NB. Remember that a fixed structured mesh is a sub-case of an adapting unstructured mesh, but our implementation of methods always assumes unstructured – this has implications on coding and CPU costs
- * Approximately: Fortran: 300k, C/C++: 50k, Python: 40K lines, including comments / white space, and in tests / examples: .flml: 160k, .geo: 17k

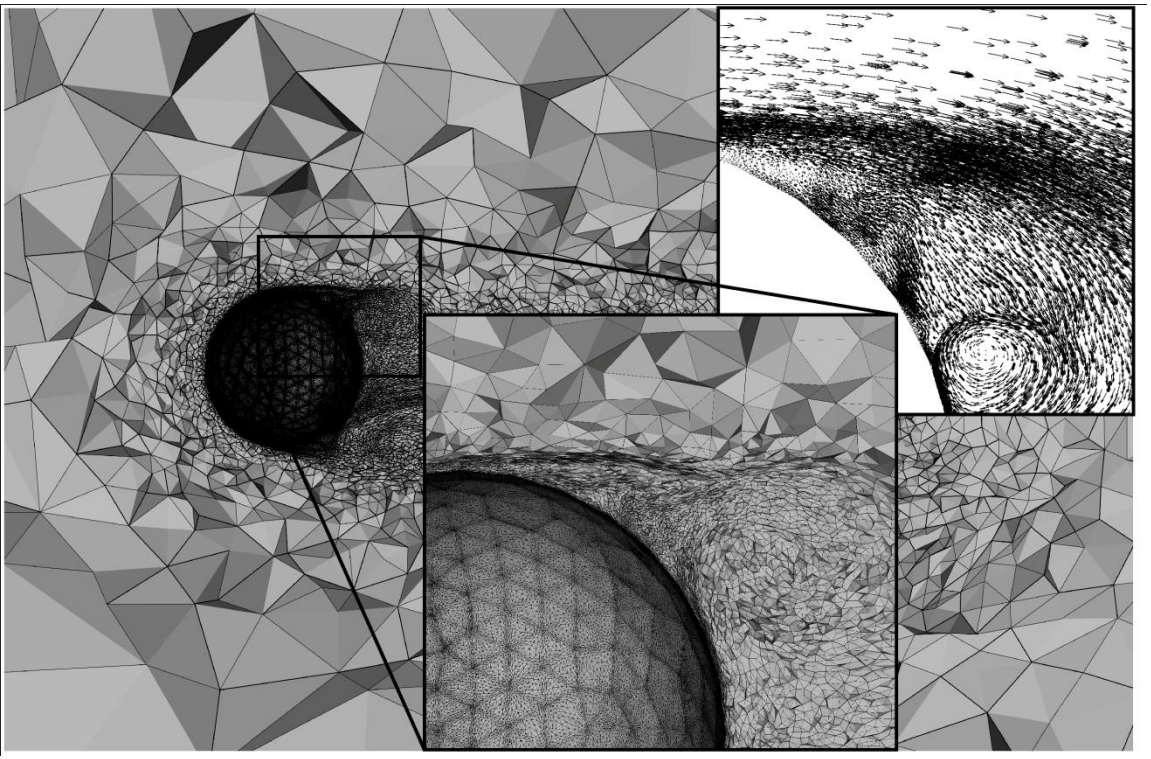
What is Fluidity?

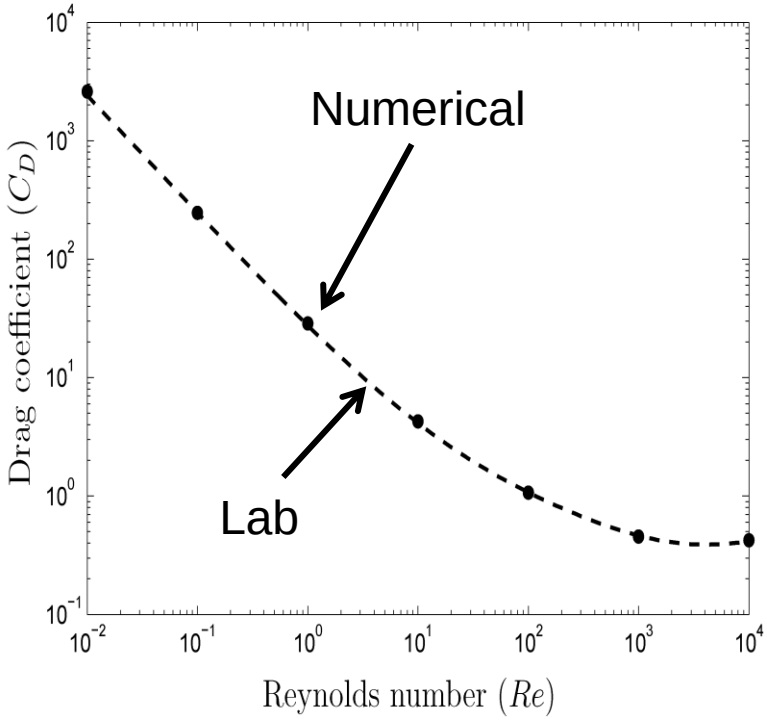
- It is (GNU LGPL) open source, and has been for a number of years
- But we are only officially releasing it and holding the first training event now
- We hope it will become a successful ‘community’ model with wide user and developer engagement and contributions

- Please remember that a code such as this is always in development
- We need you input and comments on all aspects: technical, pedagogical, etc...
- In particular, one role of this first training workshop is to see what works and what doesn’t, please send me your comments and suggestions

Some computational fluid dynamics (CFD) examples

CFD validation: Comparisons between drag calculation in flow past a sphere at a range of Re numbers



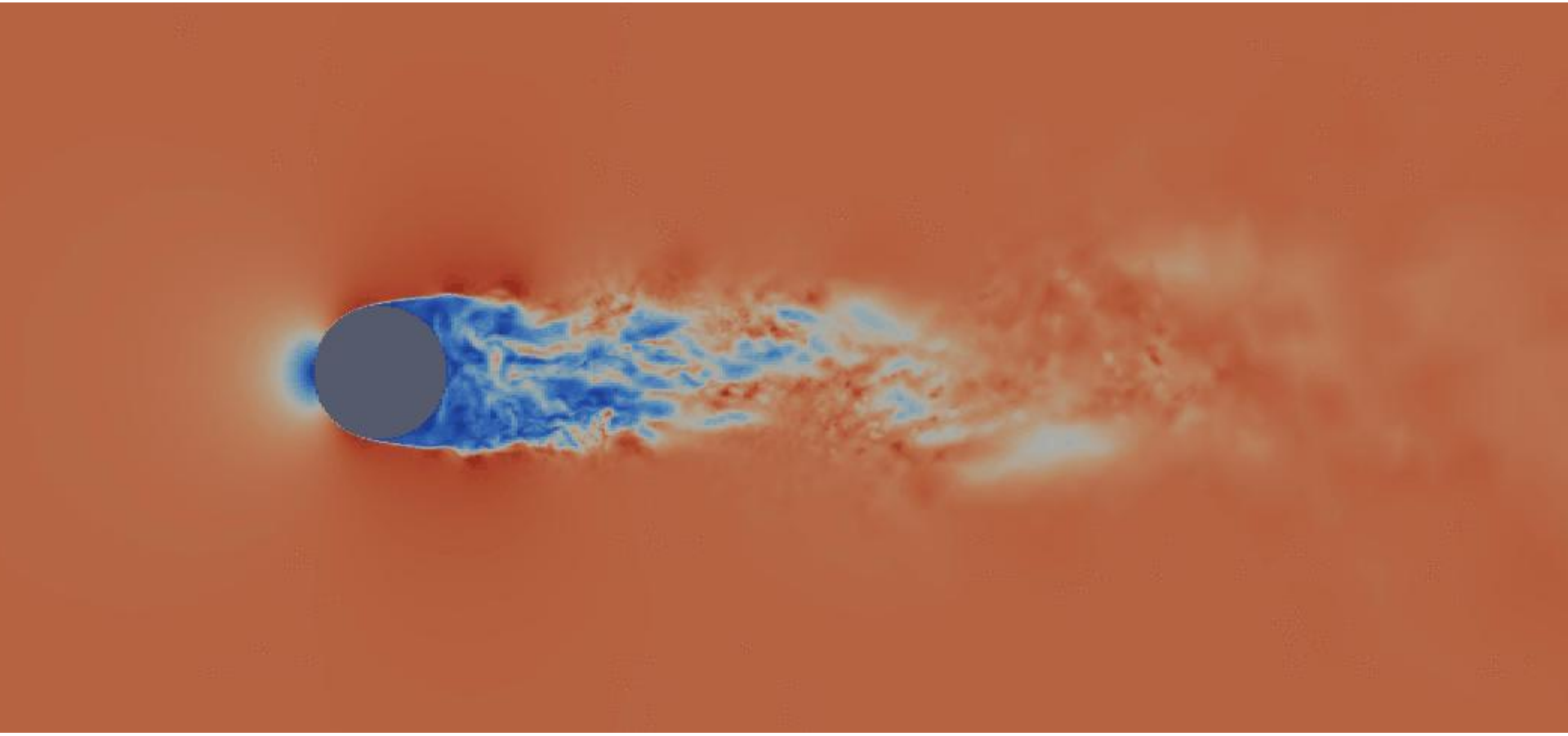


Computed drag coefficient compared against correlation (from Brown and Lawler, 2003) with lab data

$$C_D = \frac{24}{Re}\left(1 + 0.15Re^{0.681}\right) + \frac{0.407}{1 + \frac{8710}{Re}}$$

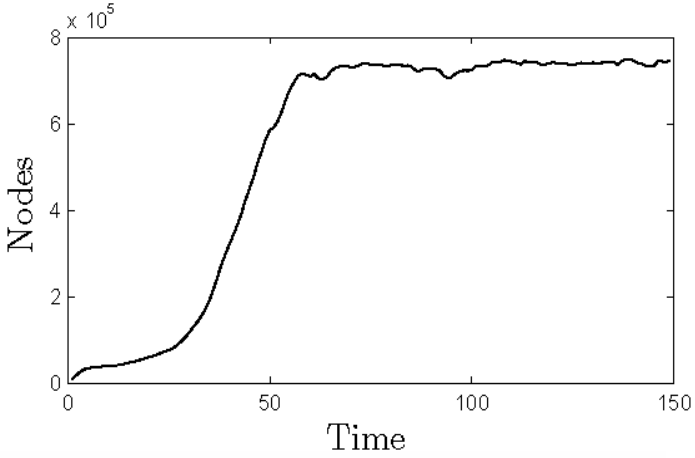
A movie showing the magnitude of velocity in a slice through the 3D domain

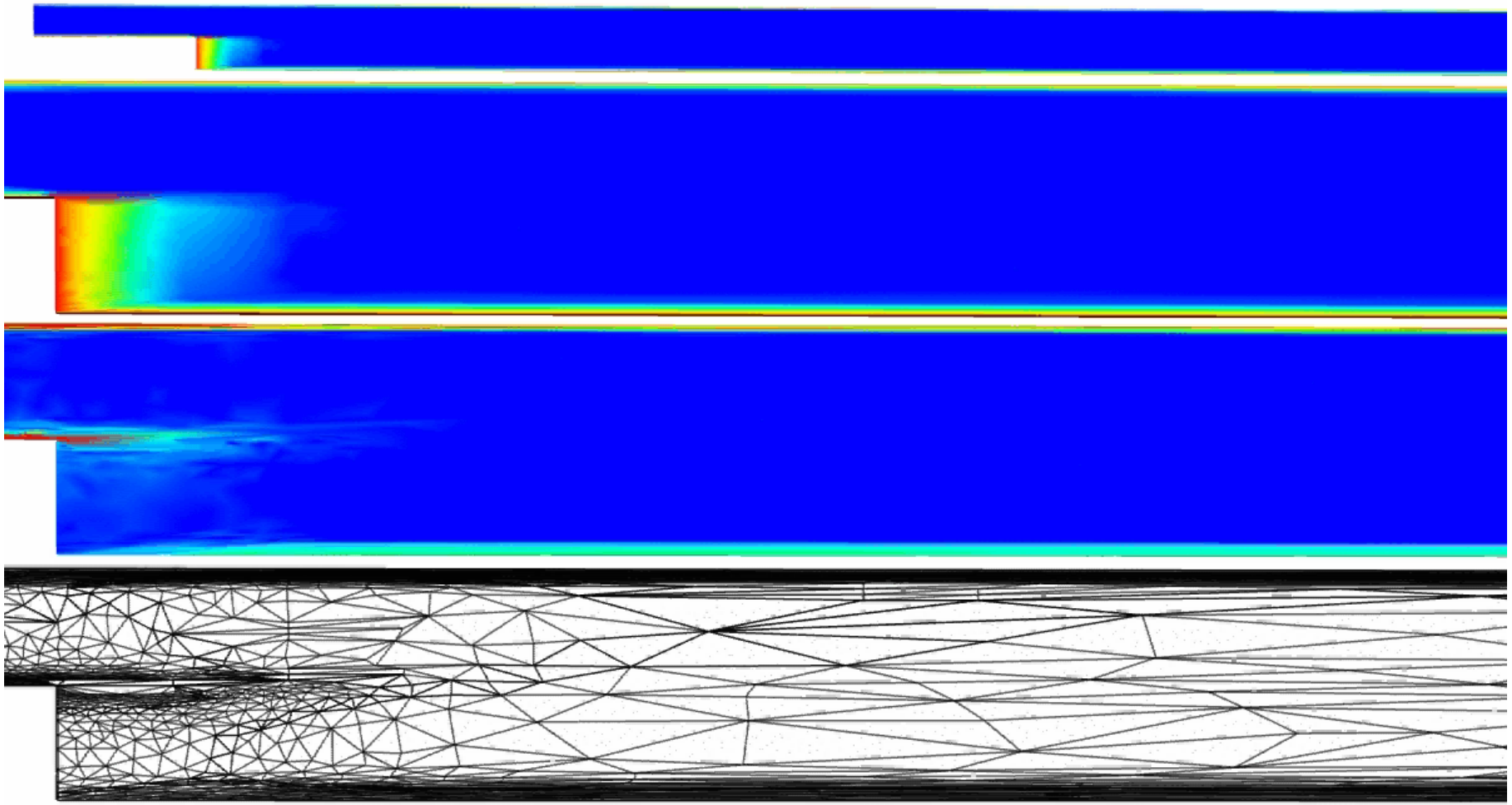
Notice that the problem is using an adaptive mesh and with the settings used the wake becomes under-resolved downstream of the sphere



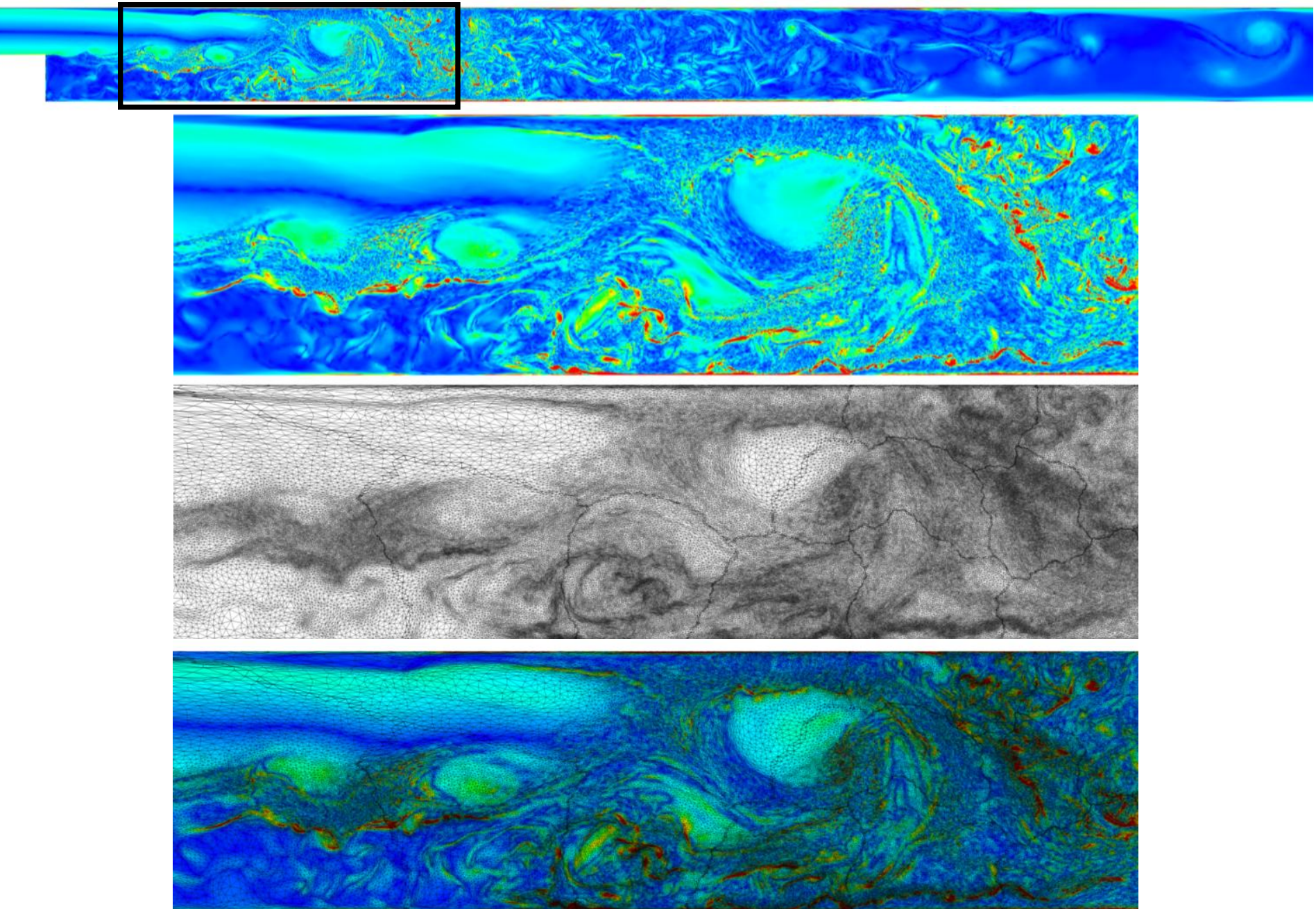
CFD example: 2D flow over a backward facing step

The whole domain, and zooms showing a passive tracer, vorticity and the adapting mesh are shown



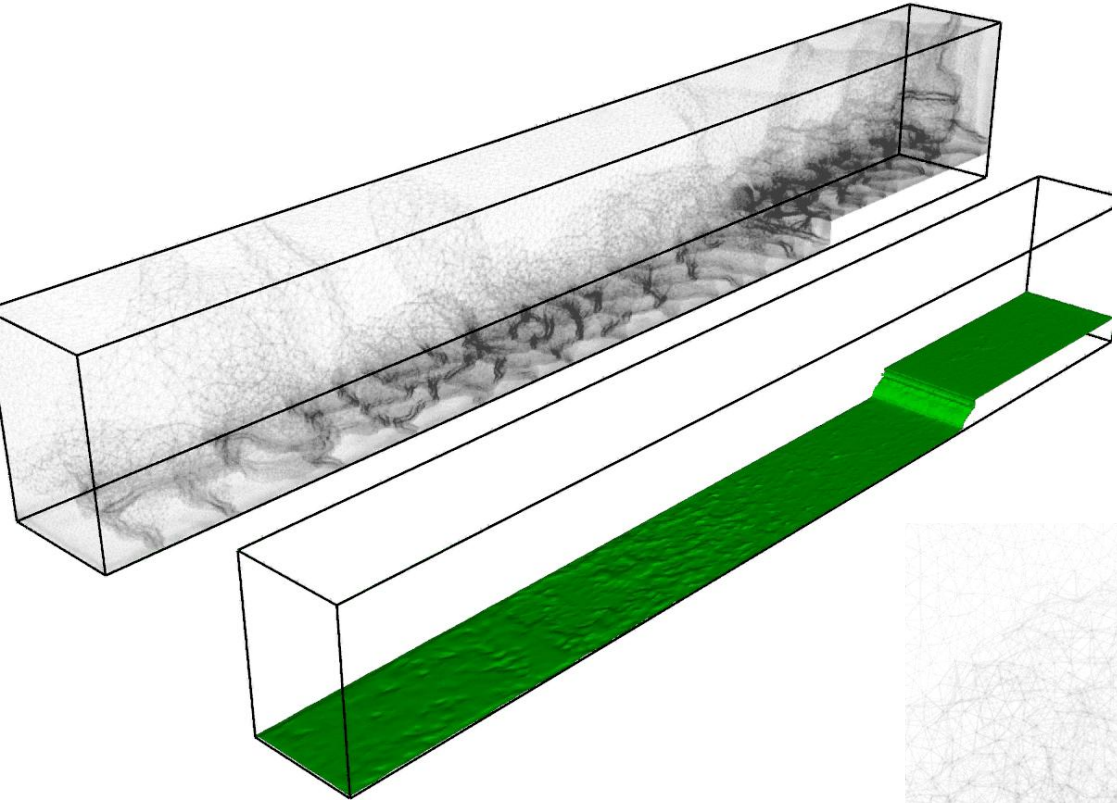


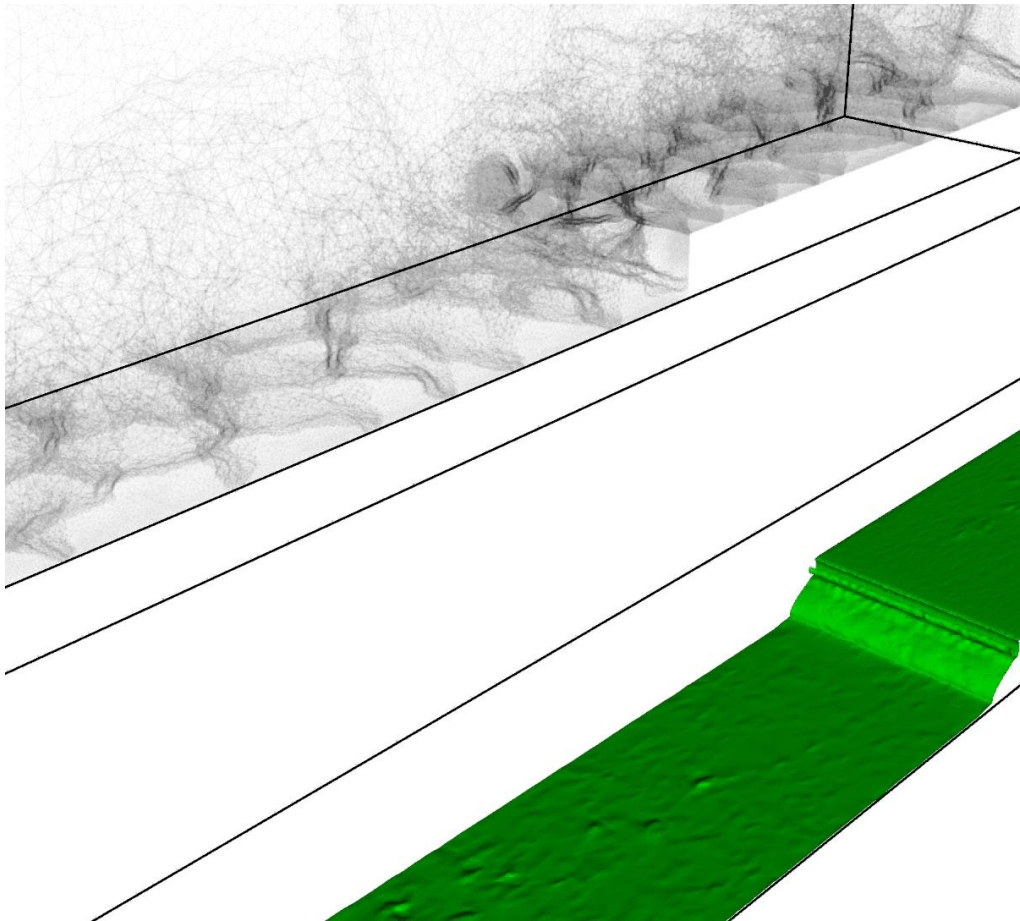
An example of the complex structures that can be resolved



3D backward facing step

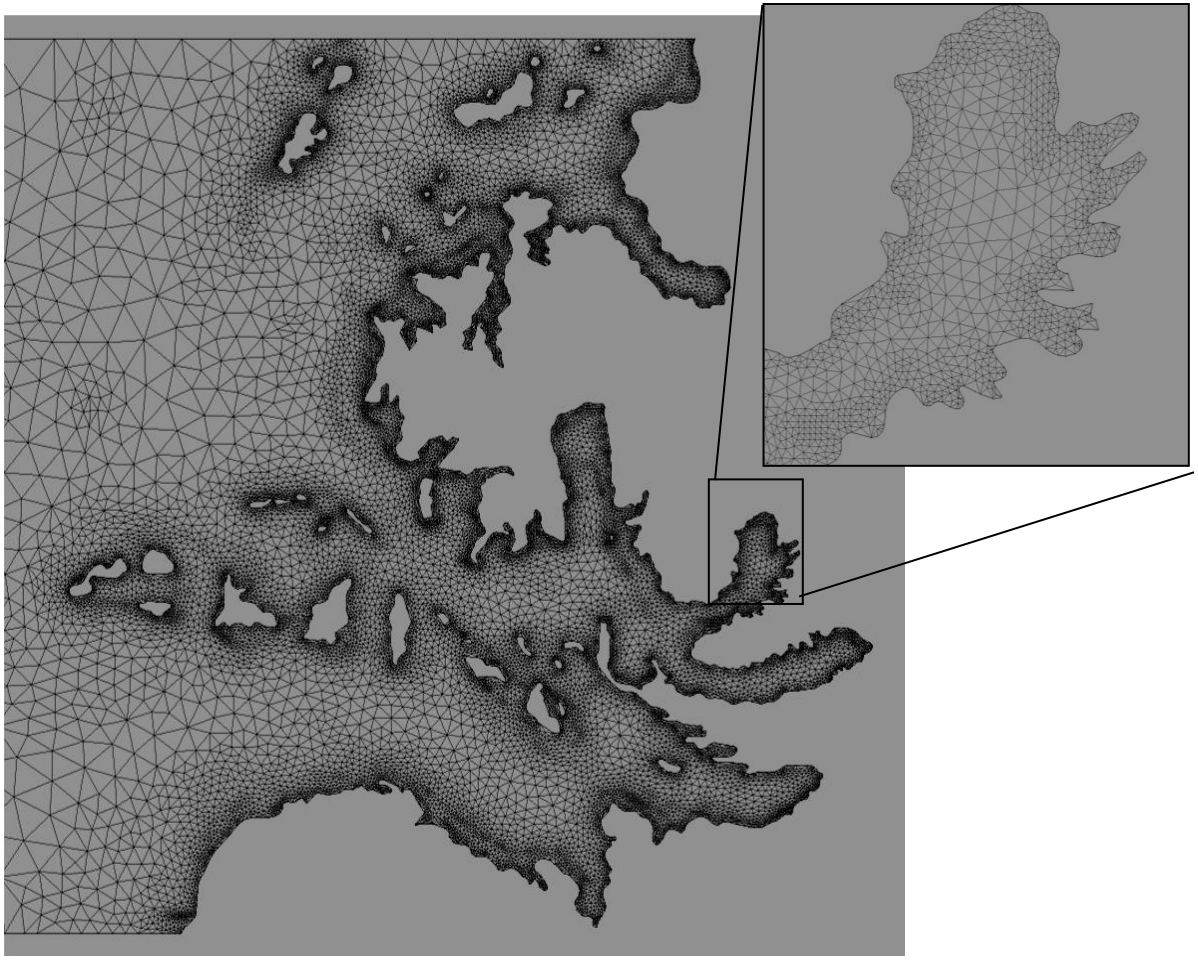
This is run in parallel and the individual sub-domains can be seen

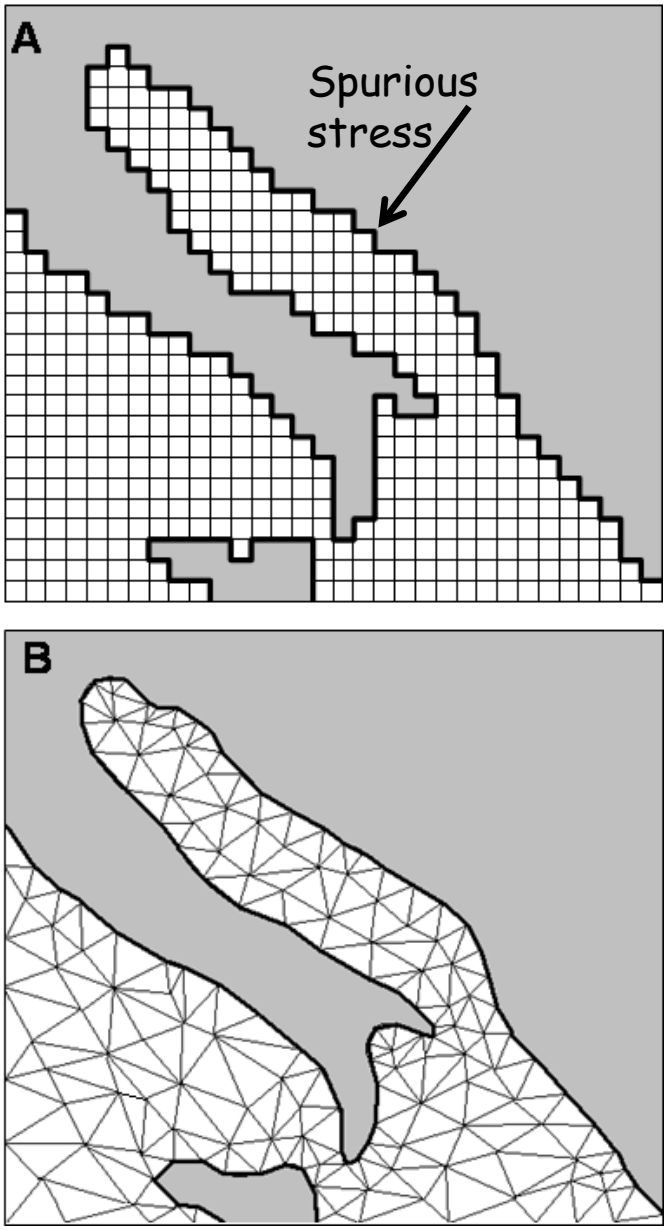




Structured vs unstructured meshes

Better representation of geometry, also the ability to the subdivide elements (hence smoothly varying resolution) without leaving ‘hanging nodes’



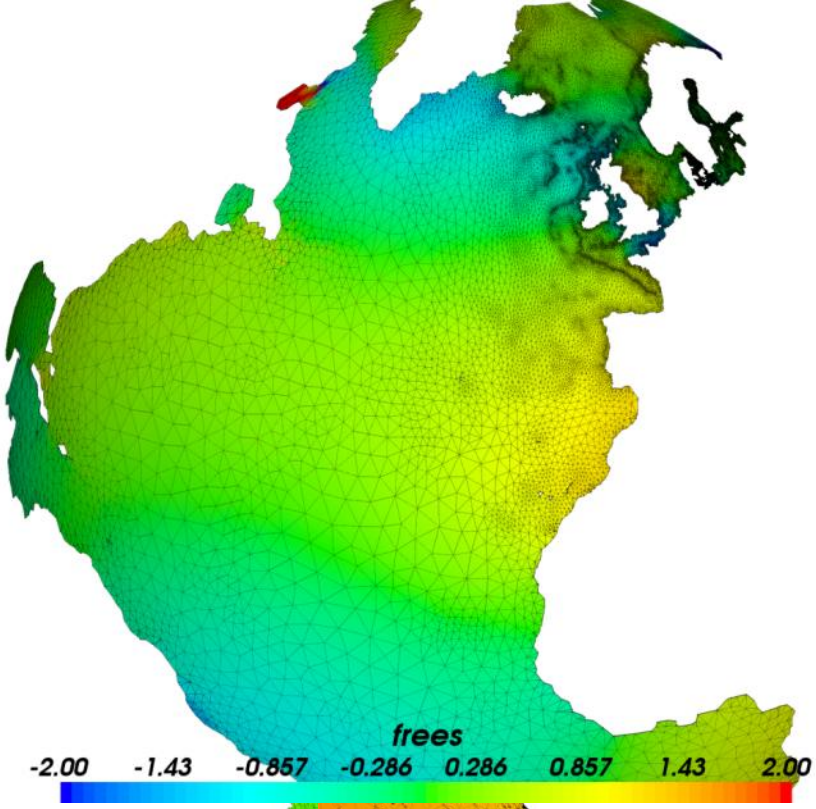


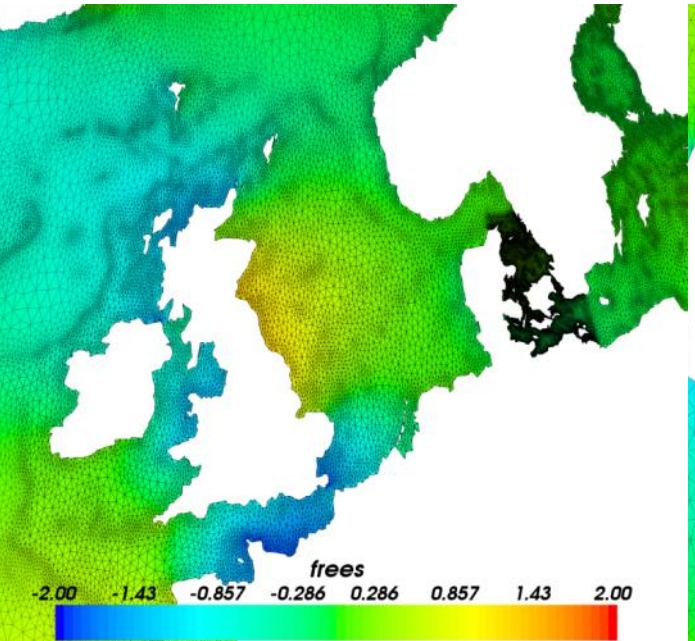
Some geophysical fluid dynamics (GFD) examples

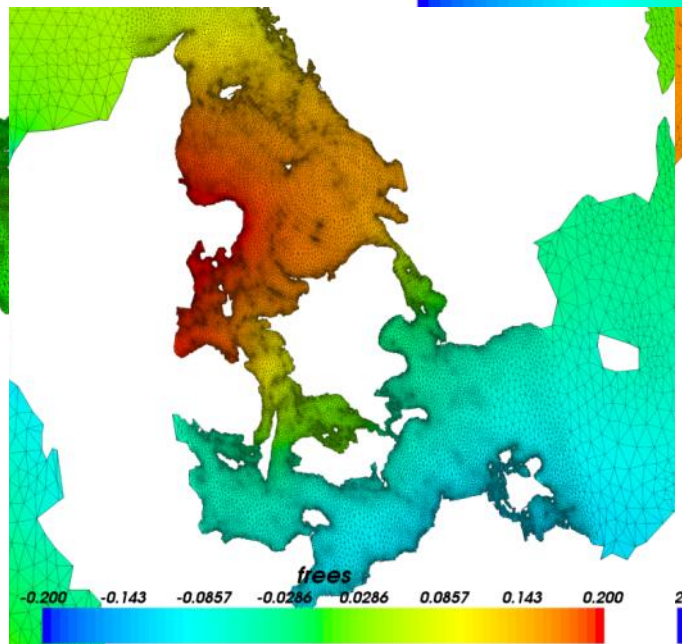
A highly multi-scale example: astronomically forced M2 tide in the North Atlantic coupled to the North and Baltic Seas

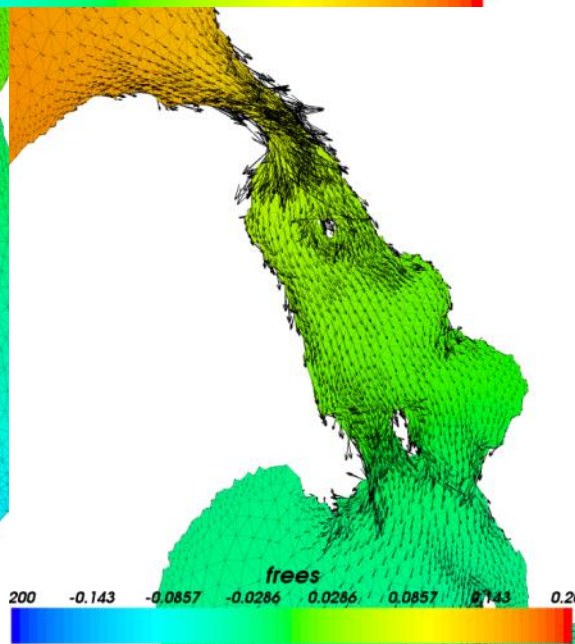
A high resolution mesh is required to resolve the channels connecting the Baltic and North Seas but a much coarser mesh is acceptable in the North Atlantic

More than two orders of magnitude difference between coarsest and finest resolutions

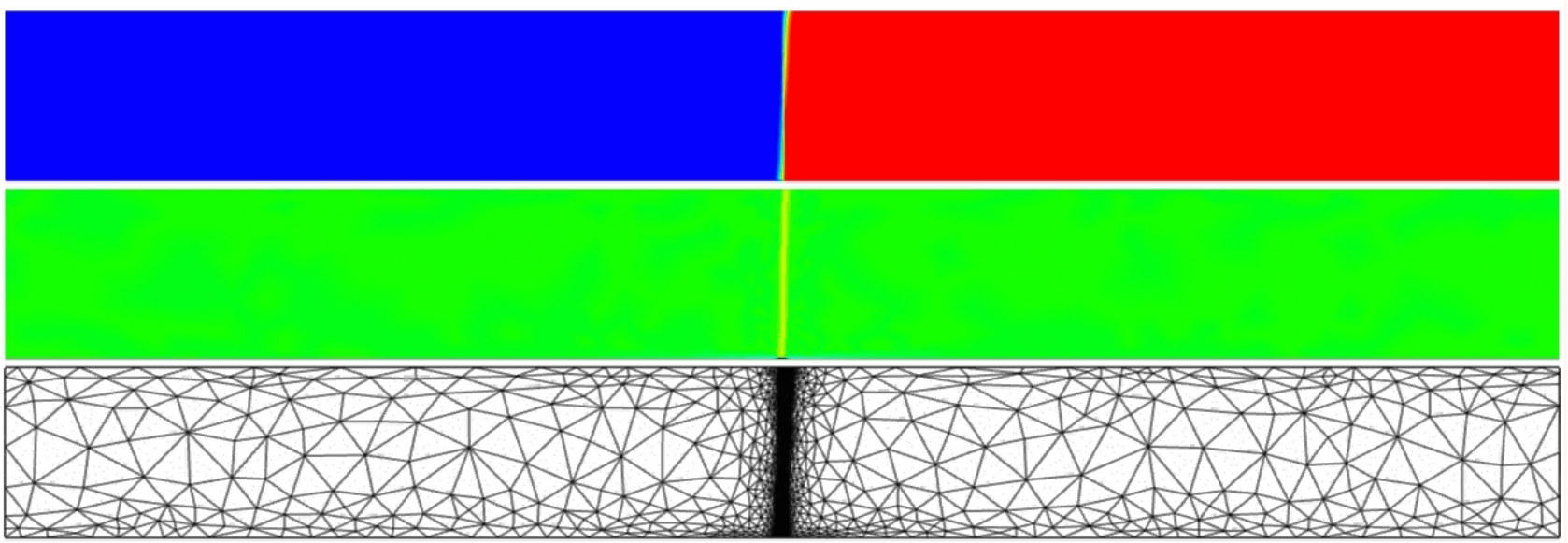


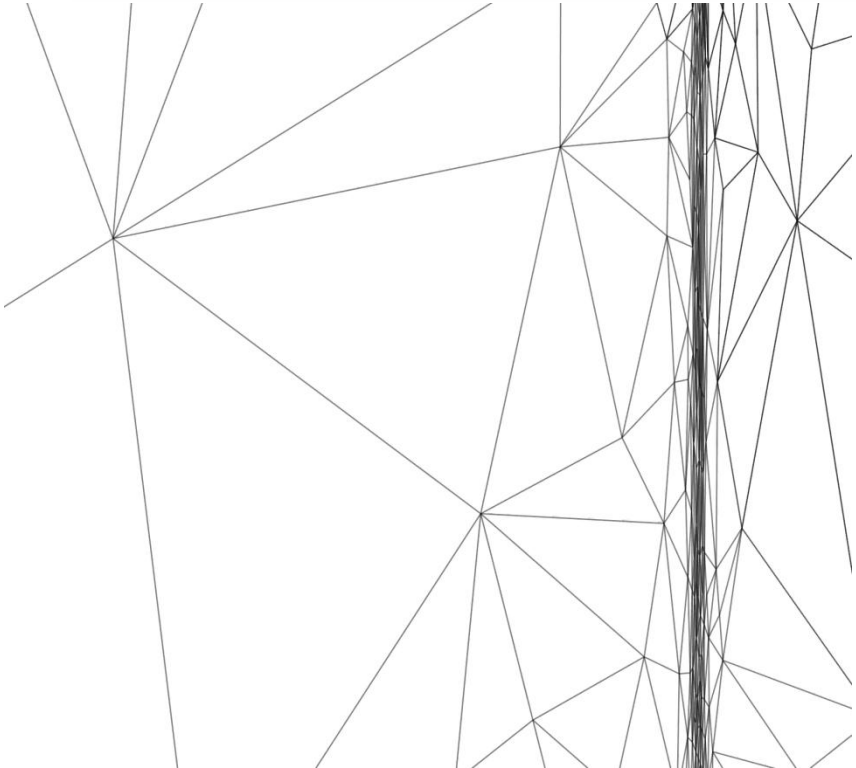


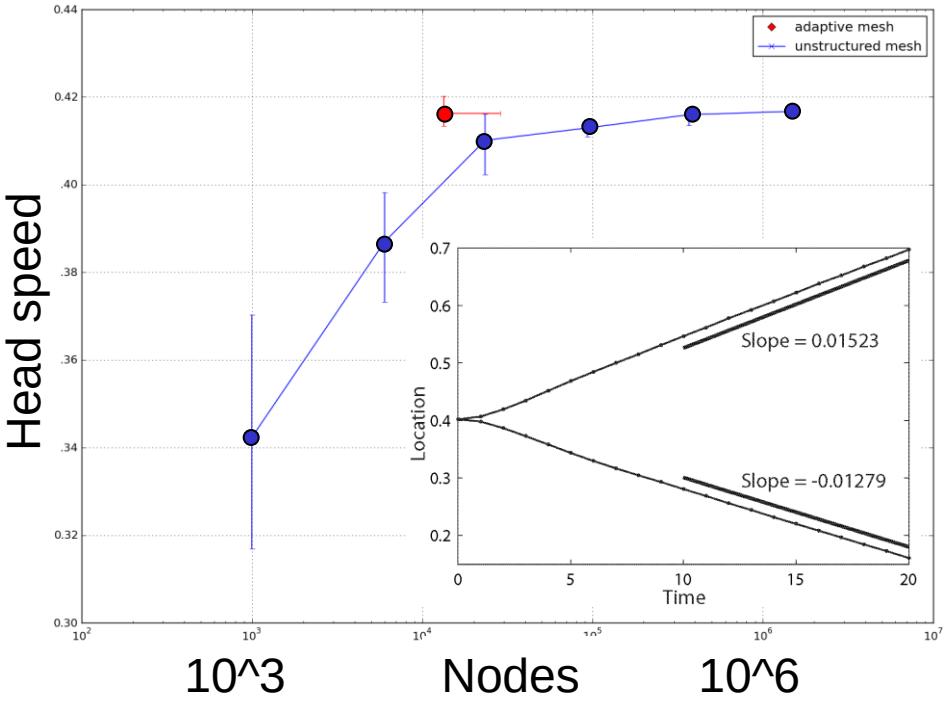




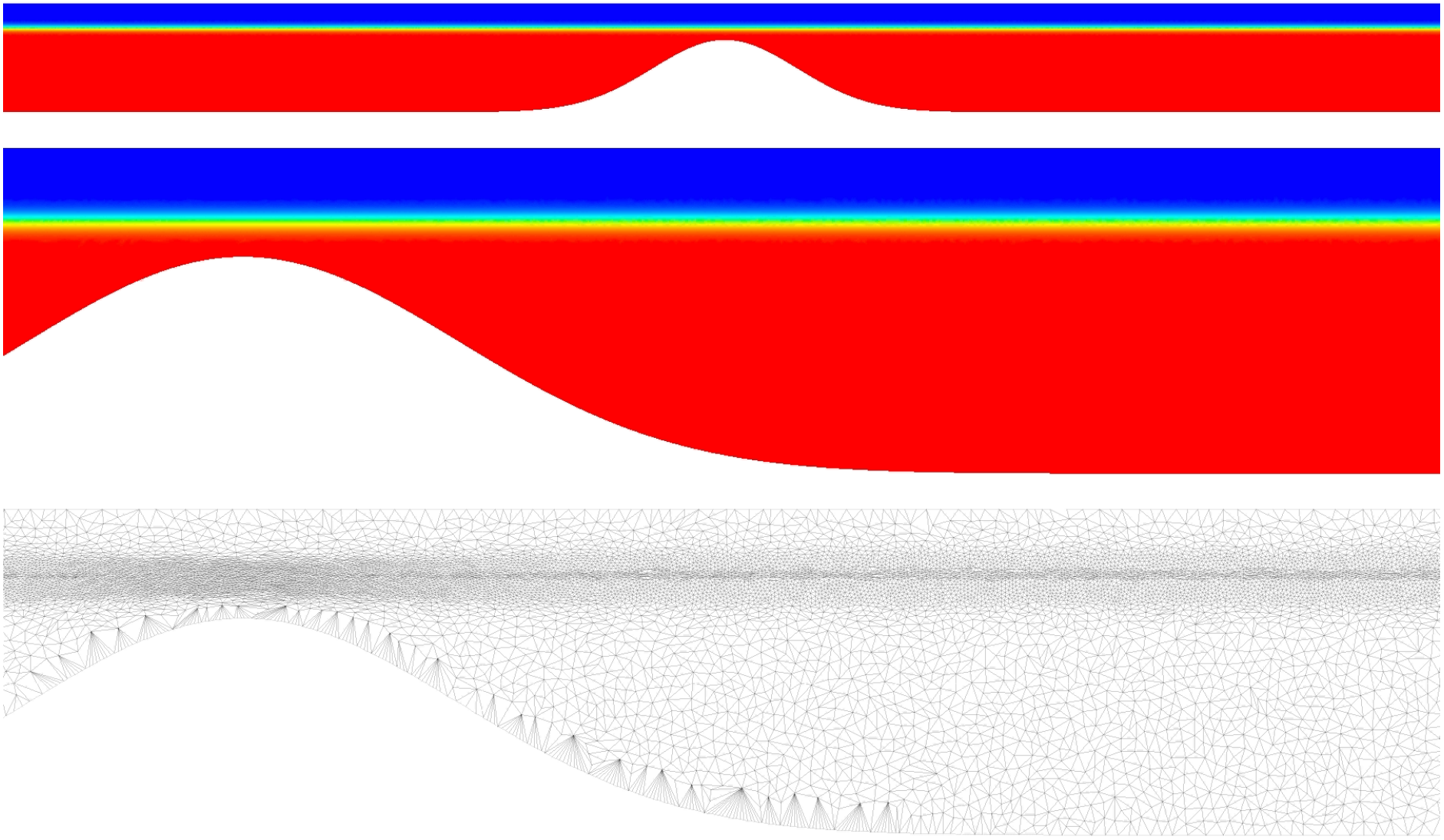
2D example with buoyancy: lock exchange problem



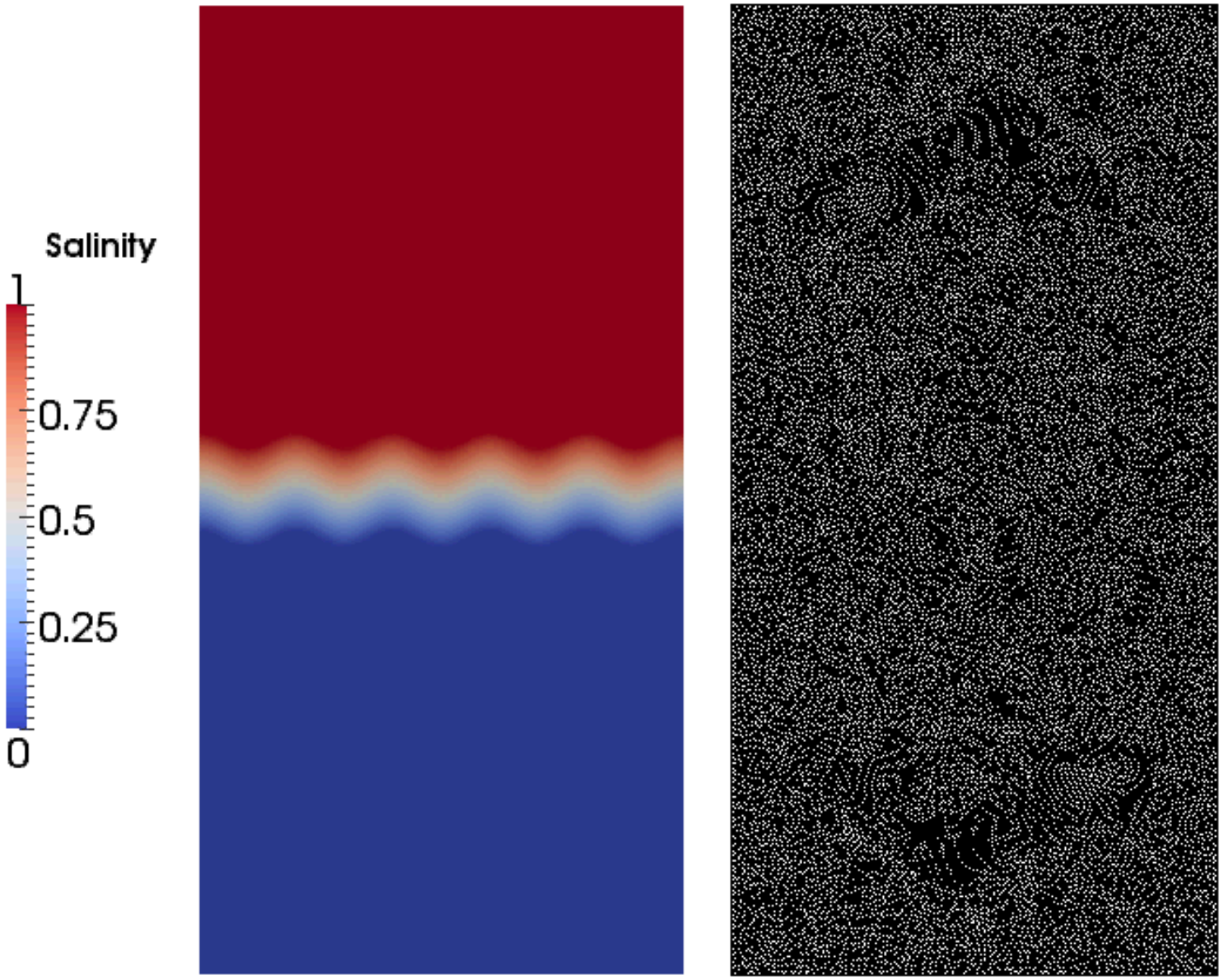




Here the adaptive mesh is focusing in and preserving the integrity of the thermocline in a lab scale simulation of internal wave generation, breaking and mixing over bathymetry

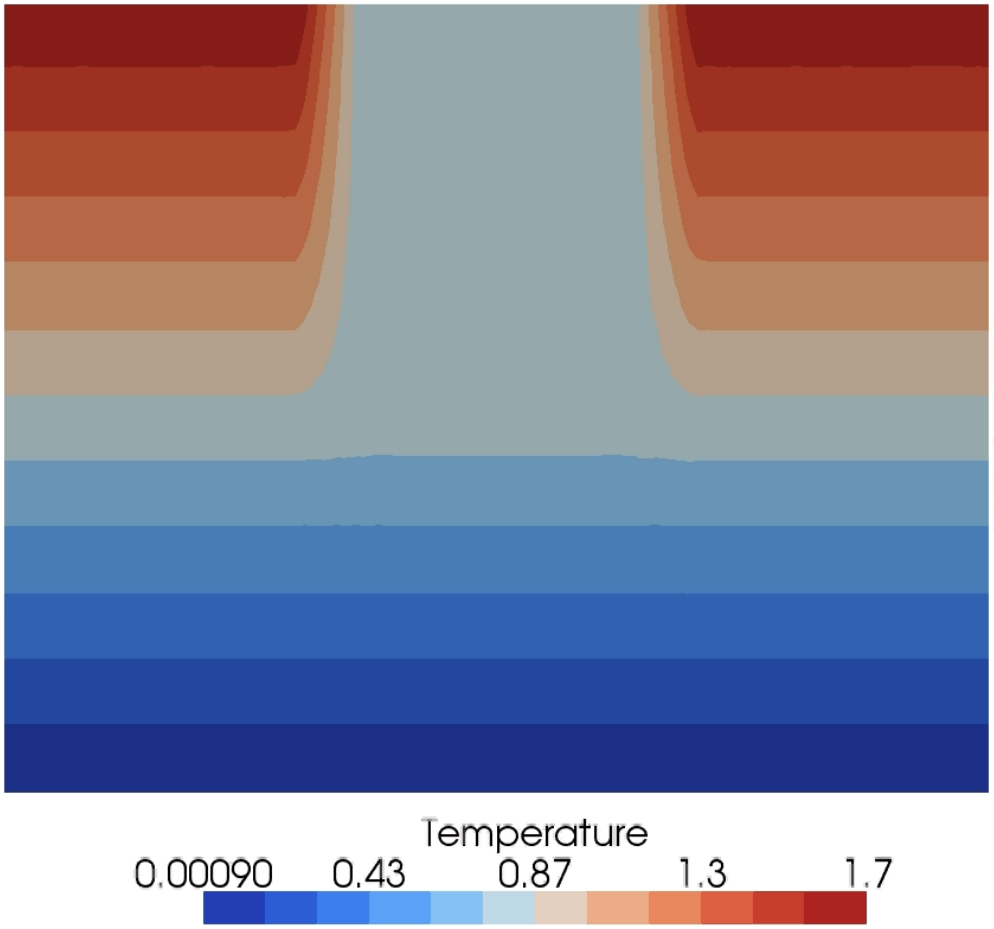


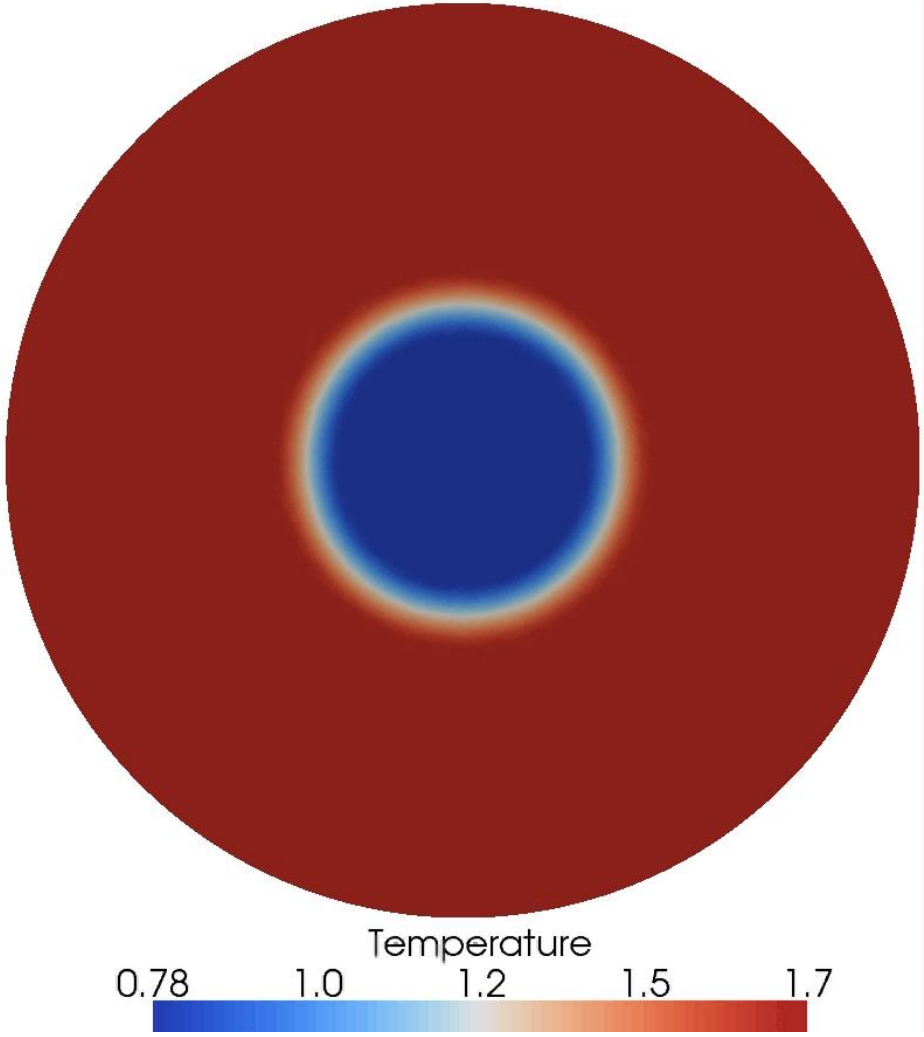
Salt fingers: double diffusive process in a periodic domain



A high* aspect ratio restratification problem with Coriolis and buoyancy

The real aspect ratio 500:1





Set up taken from: Rousset et al. A multi-model study of the restratification phase in an idealized convection basin. Ocean Modelling, 26(3-4): 115-133, 2009.

Some introductory comments on numerical simulation

What should you worry about when choosing which code to use?

- Does it claim to model the sort of problem / physics you want?
- Has it demonstrated this claim?
- Do you have to pay for it?
- Does it make use of pre or post-processing software that you have access to?
- How easy is it to set-up problems from scratch, how many useful sample configurations are there?
- Is there a testing suite that ensures robustness?
- Can you see the source code, and edit / contribute if you want?
- Will you be able to compile and usefully run it on the computers you have access to?
- How active is development?
- What is the likely longevity of the project?

Some important things to consider in numerical simulation

(physical considerations)

- What is your domain / geometry
- What are the underlying physics (dominant processes, physical parameters, non-dimensional numbers, etc)
- What equations should you consider – are they available in the code currently?
- What are the boundary conditions? Are they consistent with the physics / equations
- What are the initial conditions? Are they consistent with boundary conditions, a divergence-free initial velocity etc
- Forcing – do you need an external data set or can you set via an analytical function

Some important things to consider in numerical simulation

(numerical considerations)

- Mesh – start simple, can you use a uniform, or structured mesh (and do you want to)
- Numerical discretisation (a mammoth topic)
- Do you need to ‘model’ unresolved processes or physics that is not represented (another mammoth topic)
- Is stabilisation required – use with care
- Outputs / diagnostics – are they available, are they costly and should you compute them online or offline?
- Qualitative and quantitative analysis of outputs
- Error analysis
- Mesh refinement
- Convergence (spatial and temporal)

Some important things to consider in numerical simulation

- Remember that a numerical model really is a model – it only approximates the real system, and that system may be so complex that this model may make huge assumptions / simplifications
- Remember that a single simulation is almost certainly useless,
- as it's quite possible to get pretty much the right answer for the wrong reasons
- You will want to run an appropriately chosen suite of simulations
- You definitely need to vary details of the discretisation and mesh,
- and very likely you will want to vary some of the physical parameters governing the problem
- Think about this when you are designing your simulation set-up
- You will have to be self-adaptive, e.g. based on your first simulation, and the time it takes to run, you may have to rapidly re-think things

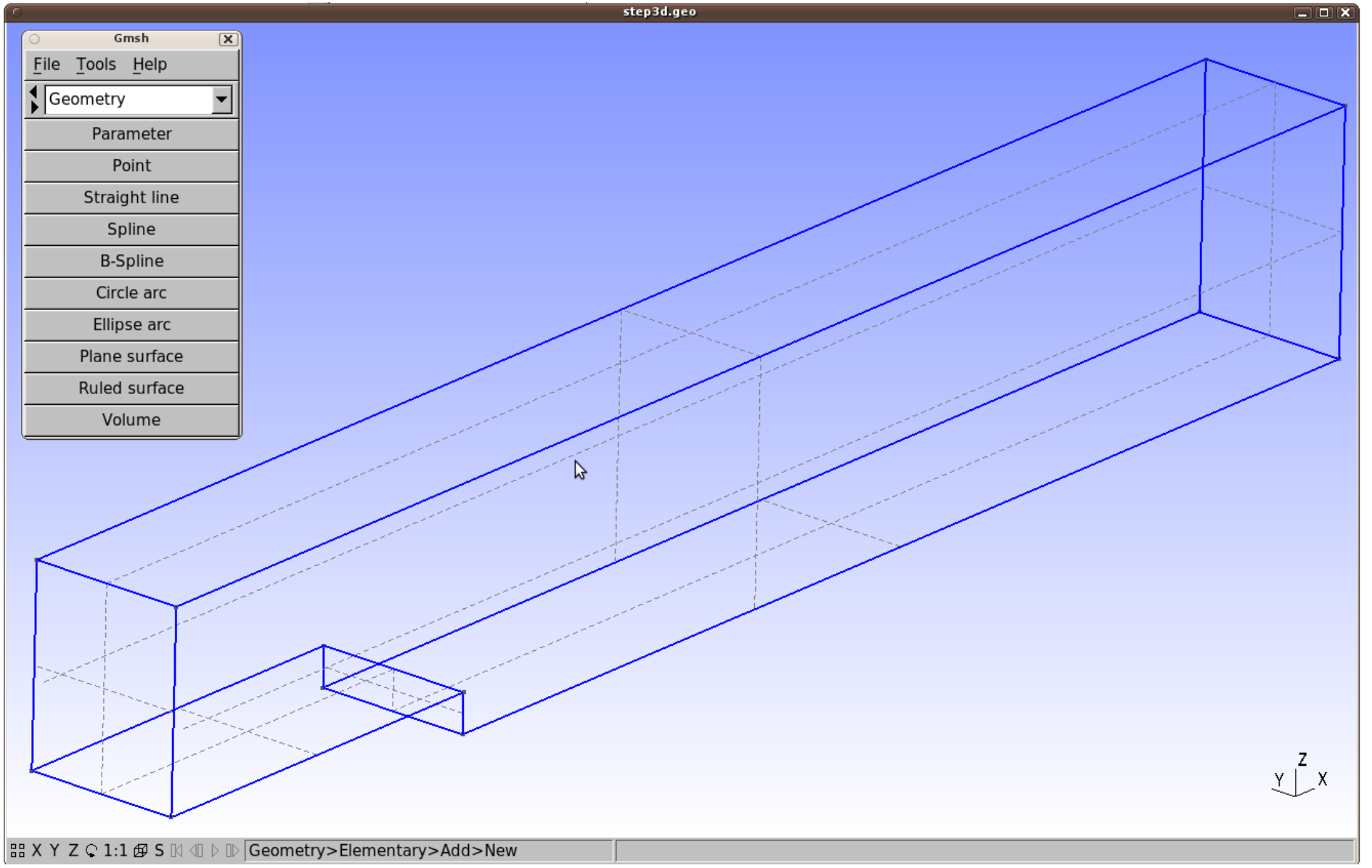
Some important things to consider in numerical simulation

- In presentations and publication reviews you will almost certainly be asked questions such as:

- “How do you know that your results are correct?”
- “Are your simulations, and hence your conclusions, sufficiently independent of the details of the discretisation / resolution?”

- So think about whether you have been sufficiently rigorous in your experiments (and / or are familiar with other relevant experimental studies that others have performed) to robustly respond to these

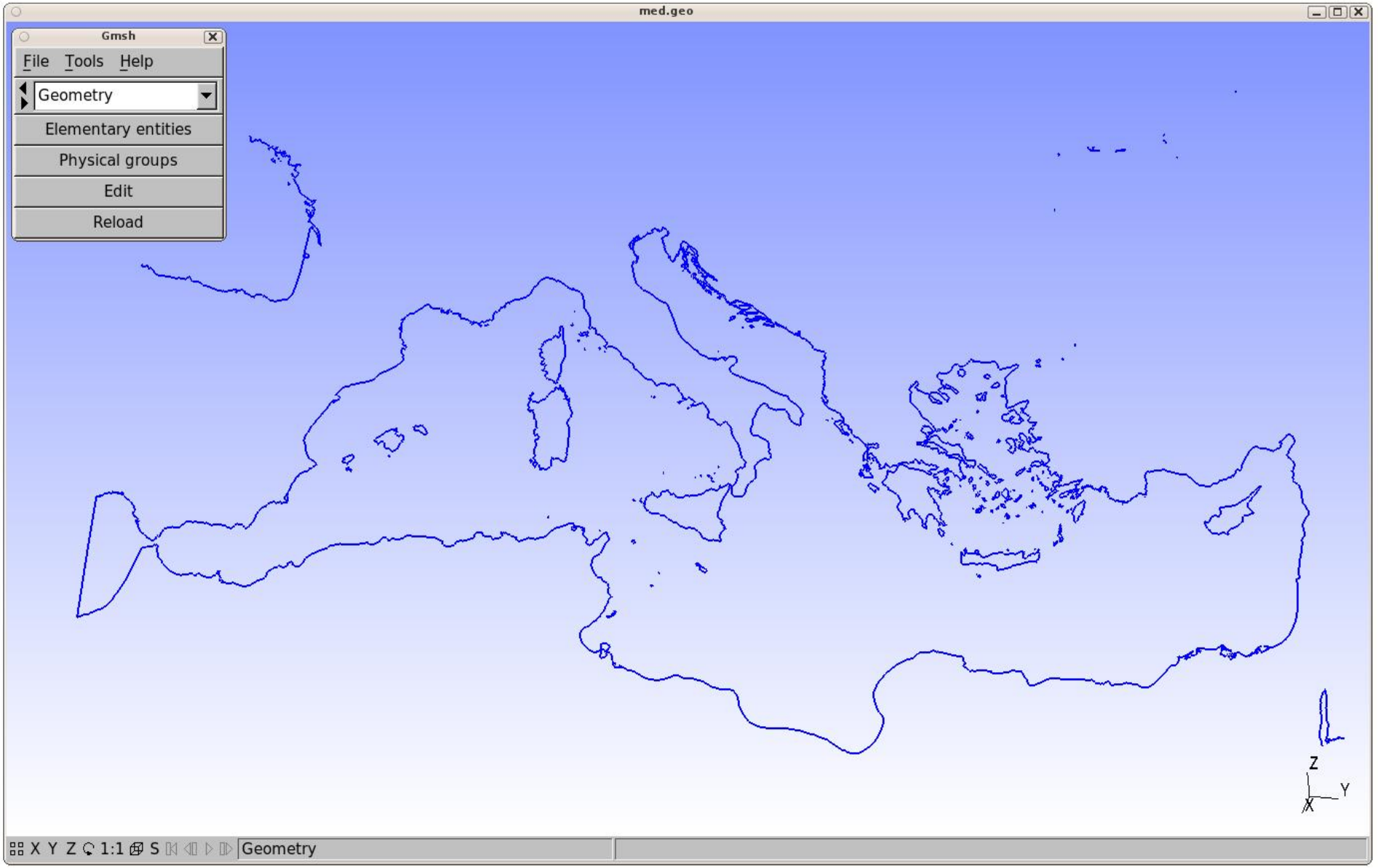
Domain / geometry – one of the CFD examples



Domain / geometry

- Obviously this is critical as it is likely to be a significant controlling factor on the solution
- Can it be defined simply, e.g. by points and straight lines in Cartesian space?
- If so, a number of mesh generators can be used to represent the surface of the domain and then mesh the interior, our current favourite is Gmsh (open source)
- If it is more complex then you may need to interface to a sophisticated CAD package, or a CAD description of the geometry may be you starting point. You may need to invest more time, effort and money in mesh generation software
- When designing the geometry you will need to think about the boundary conditions you will want to apply and consequently apply surface labels to identify them later

Domain / geometry – one of the GFD examples



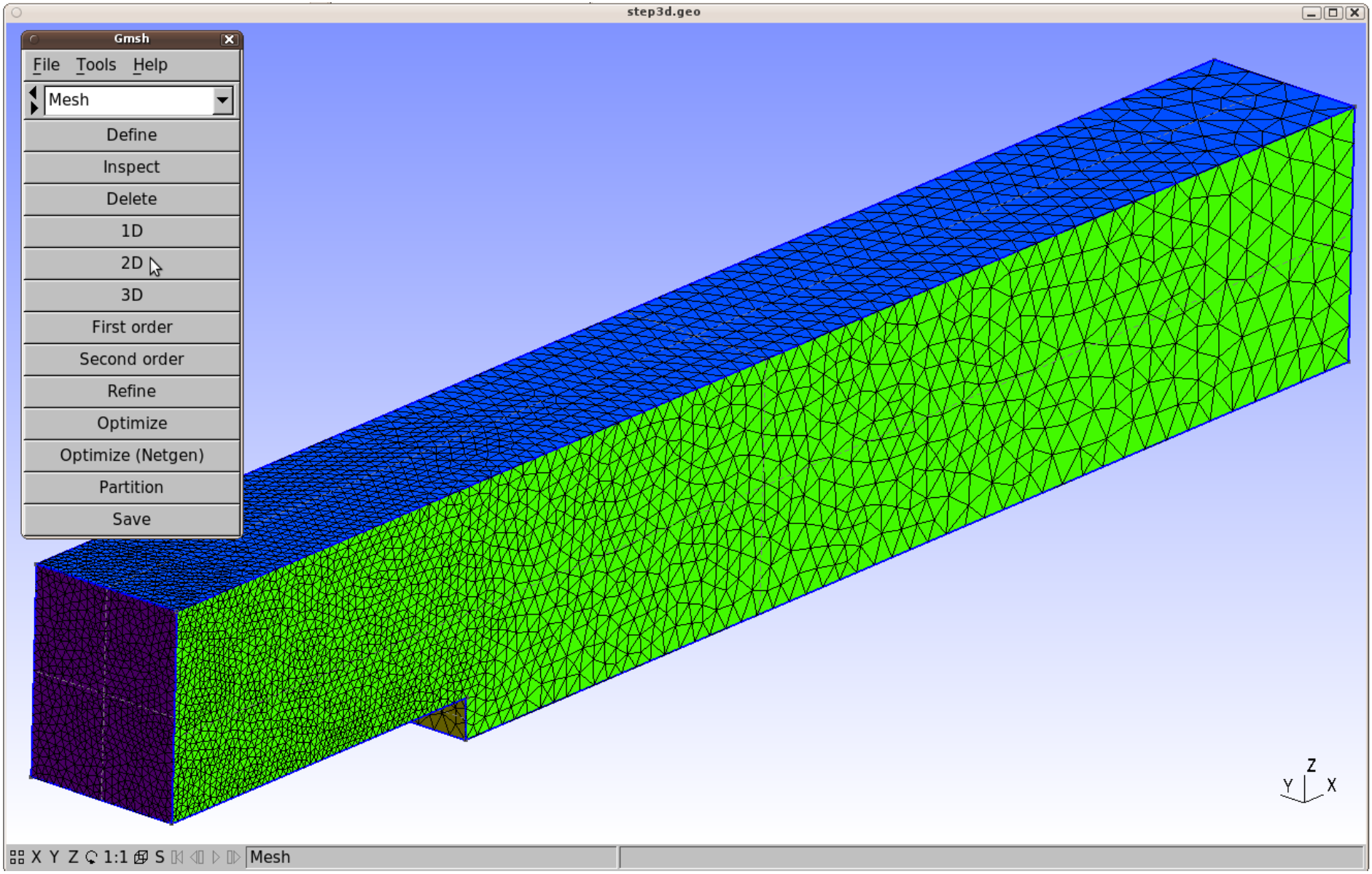
Domain / geometry – some advice

- Start as simple as possible and build up the complexity when you are confident you can run a simple problem in a simplified geometry

- Think ahead – perhaps use more labels to ID surfaces than you think you need right now, this will help you to generalise this geometry in the future

- Add comments to the file describing your geometry / surface labelling / meshing options so you can easily edit later or re-use for a similar problem

An unstructured mesh of tetrahedra for a CFD example

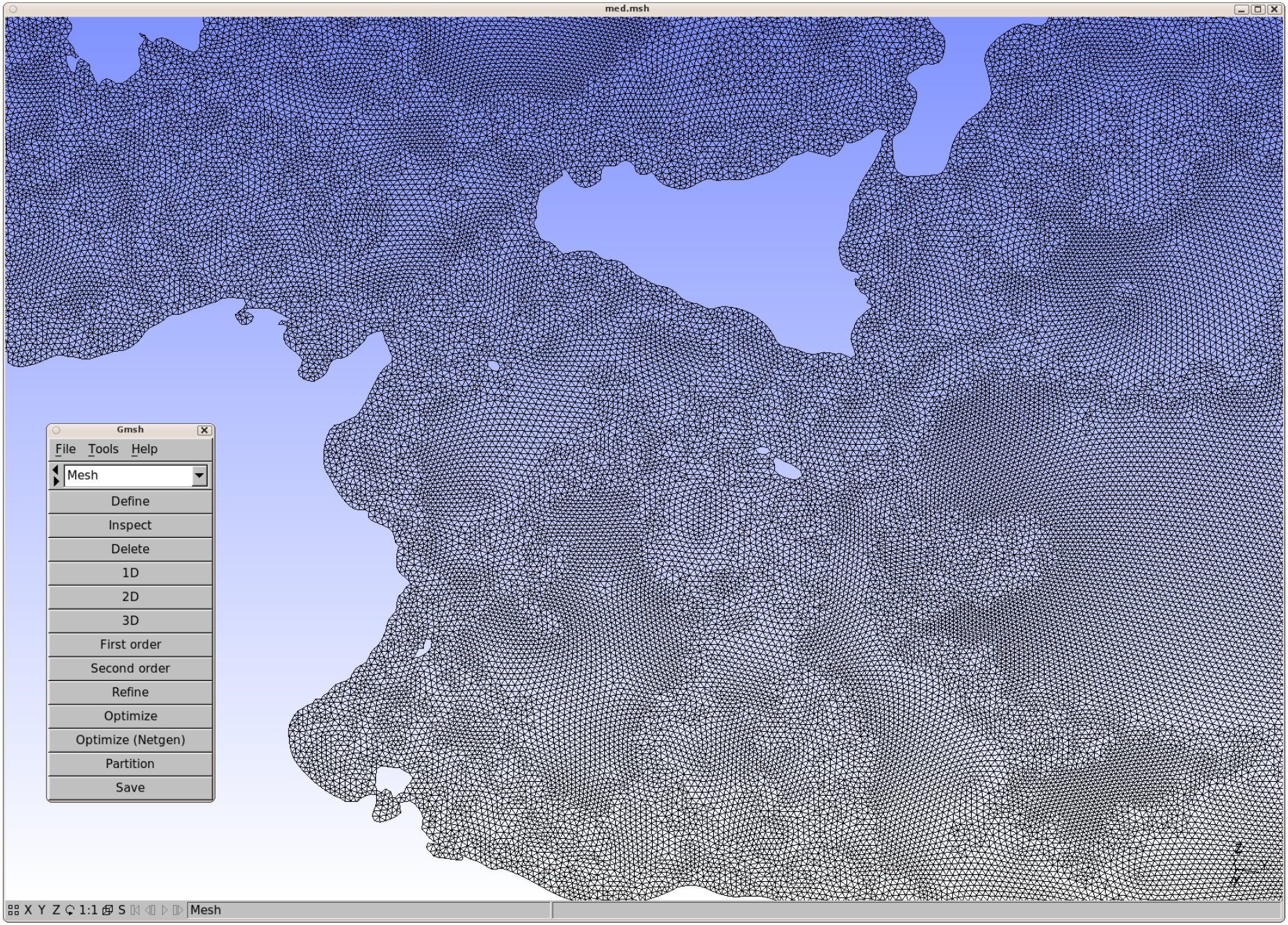


Mesh

- Define the geometry
- Mesh the surfaces
- Mesh the volume
- Do you want refinement in certain regions?
- Do you want to construct the mesh region-by-region and do you want to preserve this distinction? e.g. to preserve material properties or exactly represent initial conditions or boundary conditions
- Are you going to employ dynamic mesh adaptivity in the simulation?

- Some advice:
  - Start simple, both the geometry and mesh, and then systematically build up complexity
  - e.g. start with a box and a structured uniform mesh
  - Think ahead, choose divisions of lines so you can easily refine/coarsen later in powers of 2, e.g. starting with 2^n divisions gives you scope to both refine and coarsen in a systematic way to look at convergence

An unstructured mesh of triangles for a GFD example



Physics and equations

- What are the dominant physical processes?
- What are the unknowns?
- What equation sets do you need and how are they coupled?
- What are the physical parameters?

- Some advice:
  - Consider starting simple, e.g. possibly consider not including buoyancy or rotational effects initially and build up the complexity systematically
  - Remember that parameters such as viscosity will have a strong impact on the scales of motion – start with parameters that should yield simpler physics, and hence easier simulation, and then build up the complexity slowly

Boundary conditions (BCs)

- Don’t forget that these are just as important as the underlying equations – the problem is not completely described without appropriate BCs
- They should be based on physics, i.e. they are strongly dependent on the equations being solved and also on the geometry, not all BCs are possible / yield well-posed problems
- Many problems are strongly influenced by behaviour at boundaries and errors here are very likely to pollute the solution everywhere
- Lack of the expected order of convergence could be due to inaccuracies in BCs
- The BCs you can apply, and how they are implemented, is strongly dependent on the discretisation you are using (e.g. with FEM which terms you integrate by parts)
- We are in good shape with FEM as things match the physics naturally, but there are many options for the precise details of the BCs and so lots of scope for going wrong
- You may need to interface to external data sets, e.g. for real world atmospheric forcing

Initial conditions (ICs)

- Should be consistent with the underlying equations and boundary conditions
- Do you know the initial conditions from a previous run or an external data set?
- Do you want to define an analytical function, or interpolate from another mesh?
- Can you spin up some variables from zero? A common choice for velocity

Forcing

- External data sets
- Analytical functions
- Body forces or surface forces

- Some advice:
  - If the system is forced in multiple ways consider applying each individually before trying in combination
  - Fluctuations in the forcing could result in a particular frequency signal in the solution
  - IO / calculation could be costly

Outputs

- If you start by comparing to another numerical simulation that reports / plots a particular output metric, or a system for which some sort of observational data exists, can you reproduce these quantities for your simulation?

- Is it a costly calculation to compute this quantity – is it best to calculate online or offline?

- Run a base simulation with all the diagnostic outputs you want, and then without any, note the difference in CPU time, there could be a significant difference and this may make you far more choosy in what outputs you switch on, or to redesign an online quantity to be computed offline

- Remember that taking derivatives of the numerical solution can rapidly erode accuracy and can lead to noisy output diagnostics, that's pretty much a fact of life in numerics

- Do you need the diagnostic to be computed in a consistent manner with the discretisation, e.g. consider which function space it's appropriate to do things in

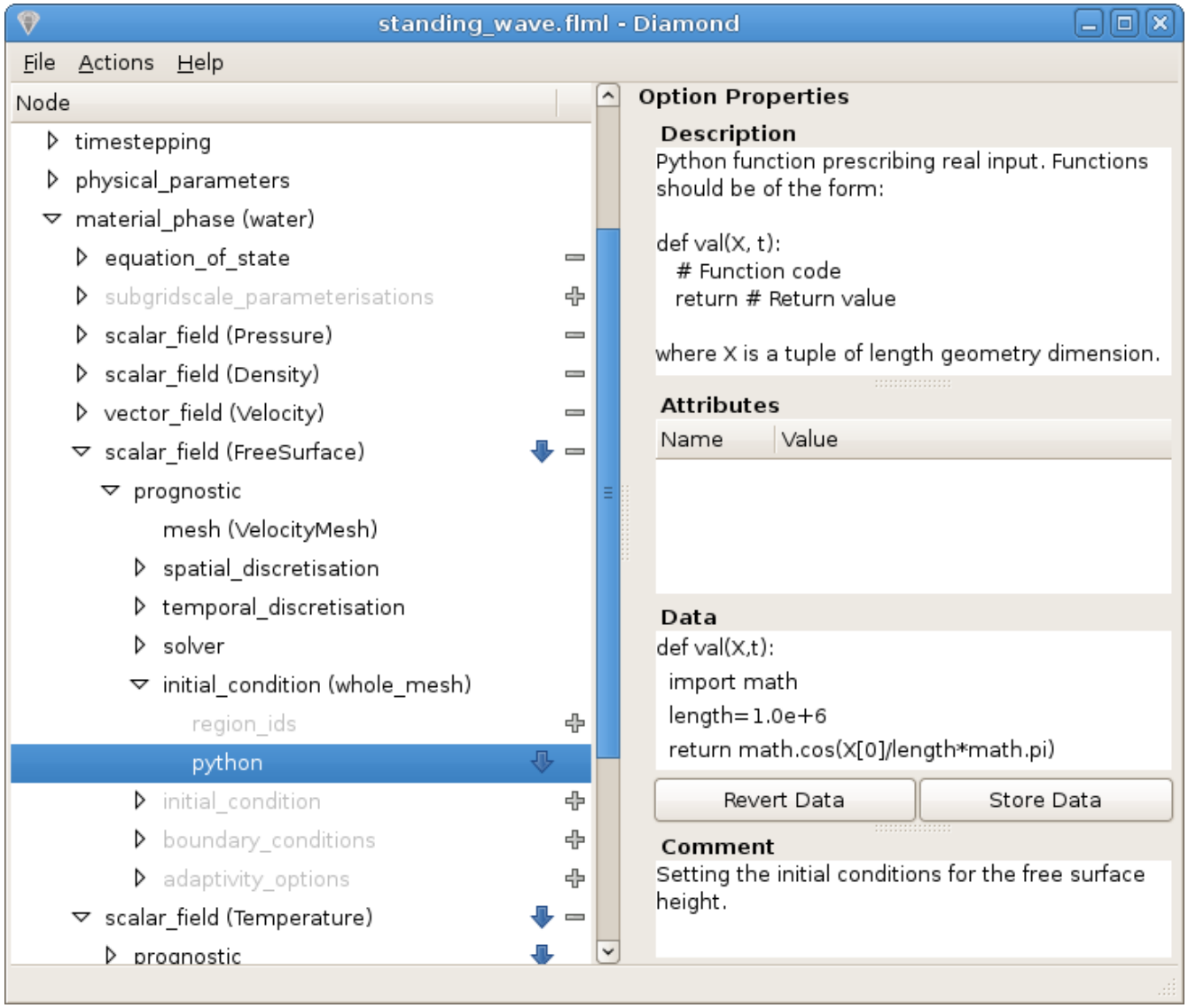
Discretisation options

- This is an entire career in itself
- Huge range of options exist
- A large number are available in Fluidity – note that it is far easier to choose a bad combination than a good one!
- The ability to make good decisions comes from a large amount of study, implementation, and experience
- Don’t forget that accuracy, stability, appropriateness of certain BCs, are all completely reliant on the discretisation choices

- Some advice:
  - Ask for advice!
  - Consider starting from a numerical configuration already set-up for a similar problem, don’t assume this is all appropriate for your problem or even for the problem it previously solved
  - Try to understand why particular choices have been made - ask lots of questions on why
  - Start defensive – a very safe / boring option with lots of stabilisation and implicitness, and then finesse once you have something working
  - Read chapter 3 of the manual, the references therein, and at the end of this presentation

Usability: problem set-up with Diamond

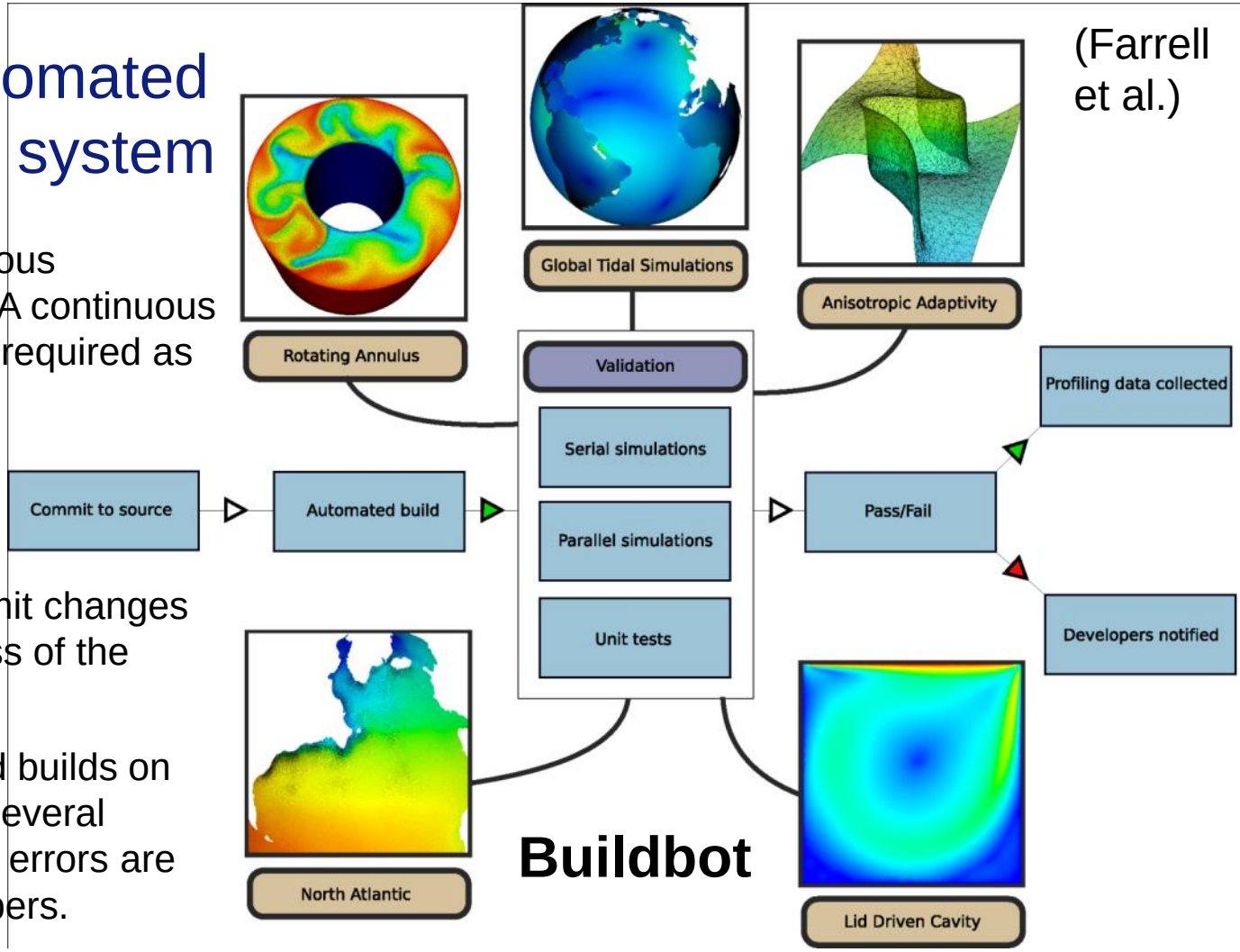
- An xml schema file describes the rules that govern model options
- Diamond uses this to automatically generate a GUI based on the schema
- Options are entered and output as another xml file containing the options values
- This is written to an options library accessible from anywhere in code
- Includes many features, including the ability to define python functions executed at run time



Robustness: automated code verification system

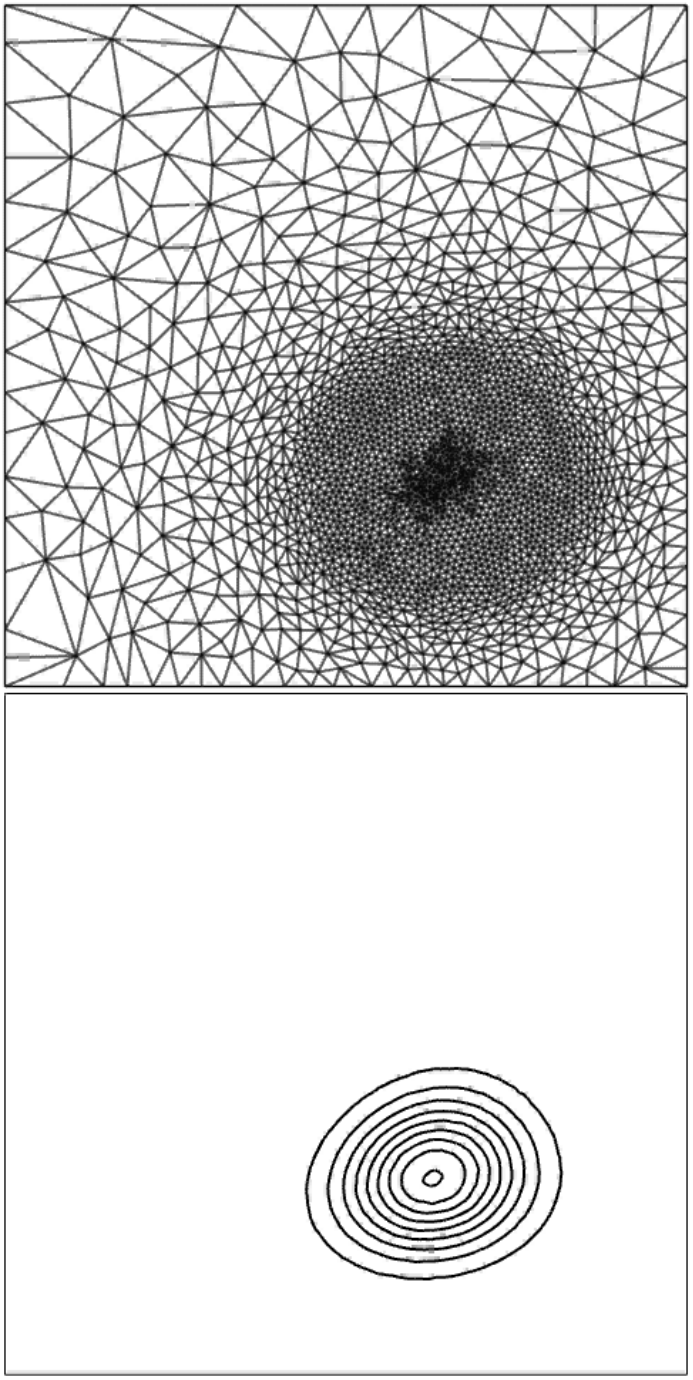
(Farrell et al.)

- All models require rigorous validation / verification. A continuous automated approach is required as the codebase changes.
- A central copy of the source is kept in a subversion repository. When developers commit changes this must result in a pass of the code test suite.
- Buildbot checks out and builds on various platforms with several different compilers. Any errors are relayed back to developers.



- If a failure is detected in a test problem, the developers are notified details via email.
- Statistical information about code quality is automatically collected from the newly validated code. This allows for the monitoring of performance. Results available via a web interface.
- This modern approach to software engineering has yielded dramatic improvements in code quality and programmer efficiency.

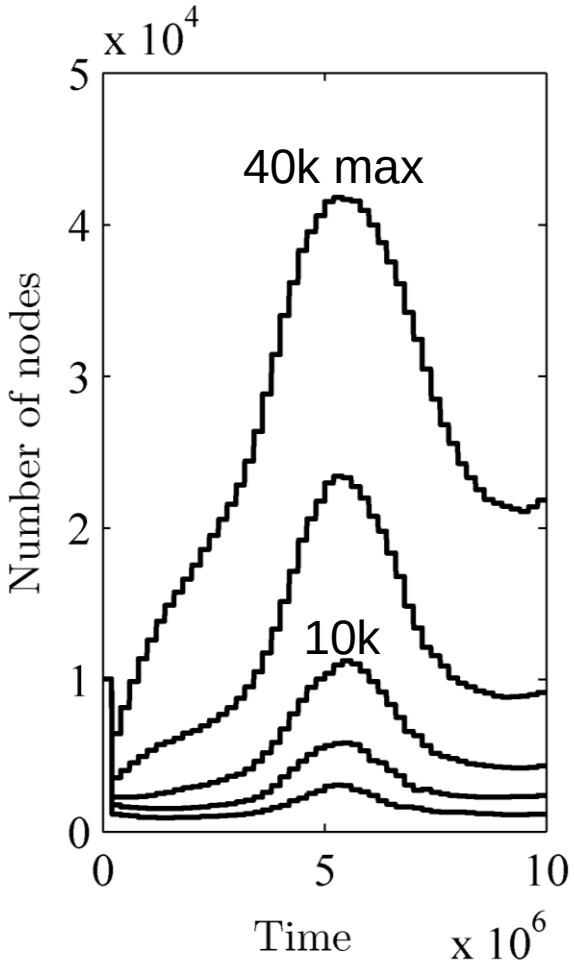
Some additional examples



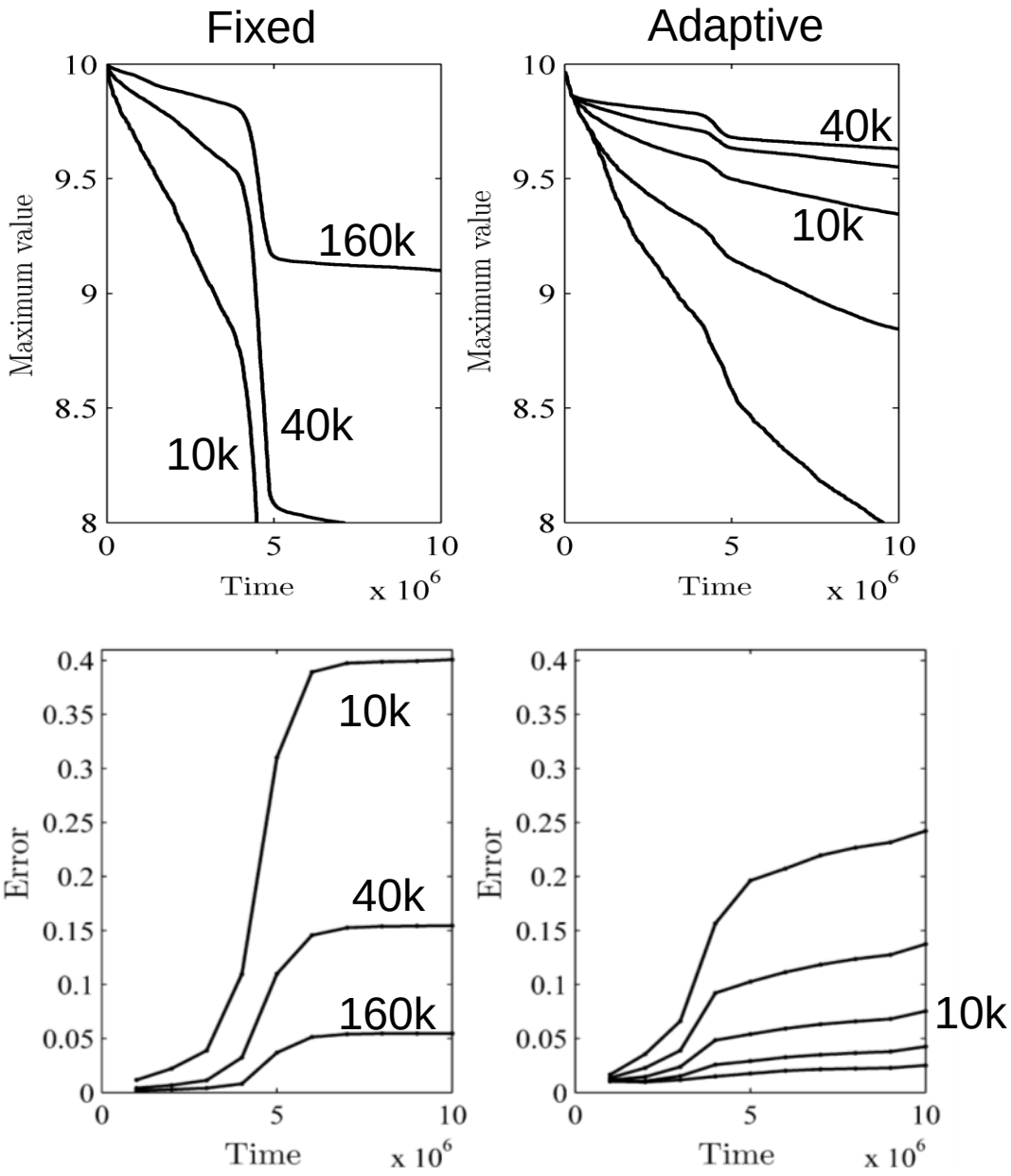
Advecting a tracer field through a boundary layer – prescribed velocity

Initial Gaussian tracer field advected with the velocity field from the Stommel gyre – stretching through the boundary layer makes this a tough problem for the advection method.

Number of nodes against time shown right with 5 different error weights.



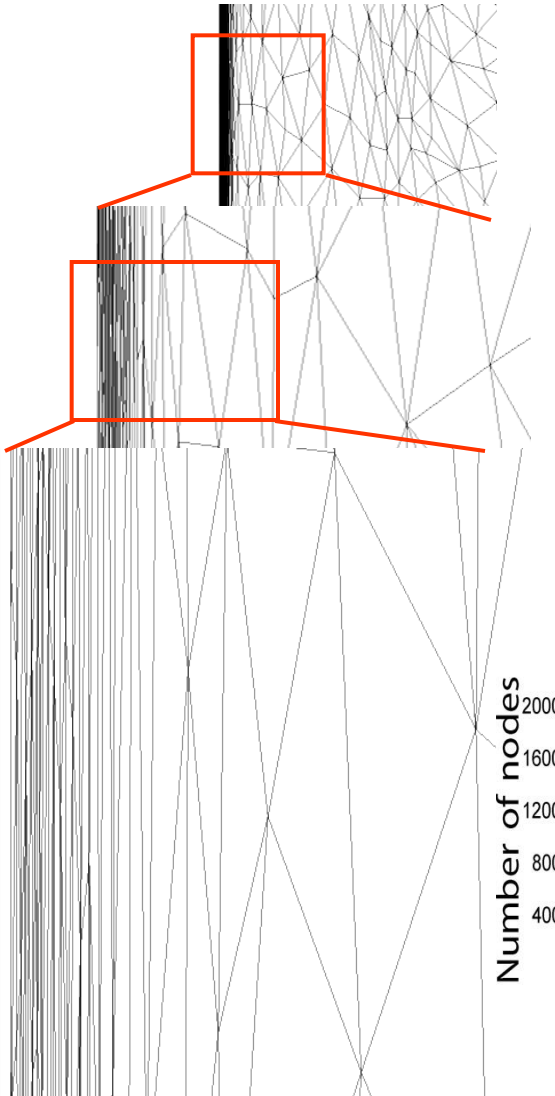
Comparisons between uniform fixed and adaptive simulations

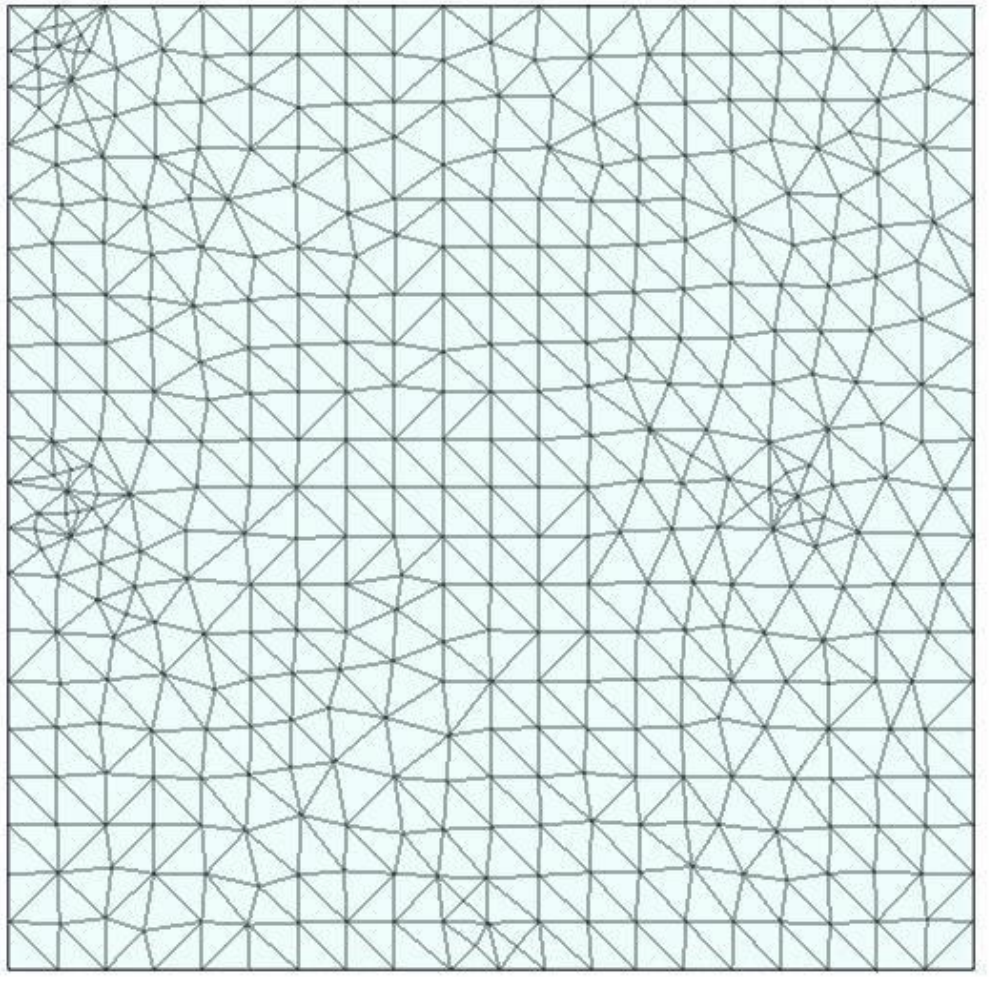


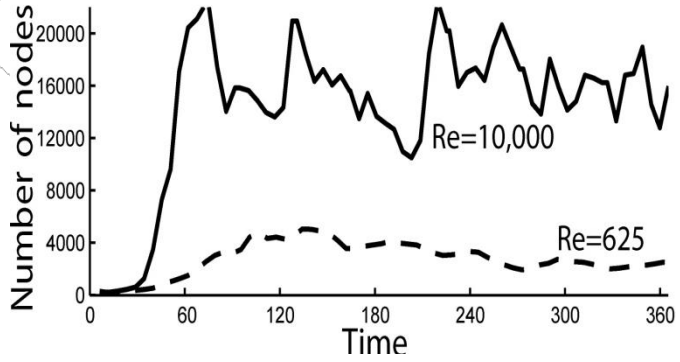
Maximum tracer concentration against time – should be constant at 10.

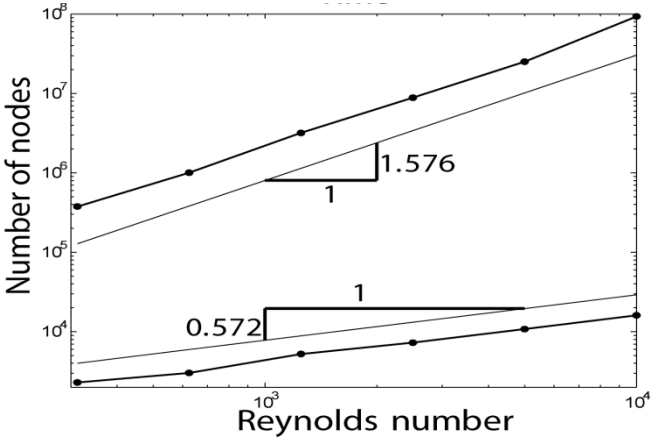
L2 norm of error compared to an exact solution obtained by integrating back particle paths, against time.

Wind driven gyre: western boundary current (e.g. Gulf Stream) and eddies resolved with anisotropic resolution



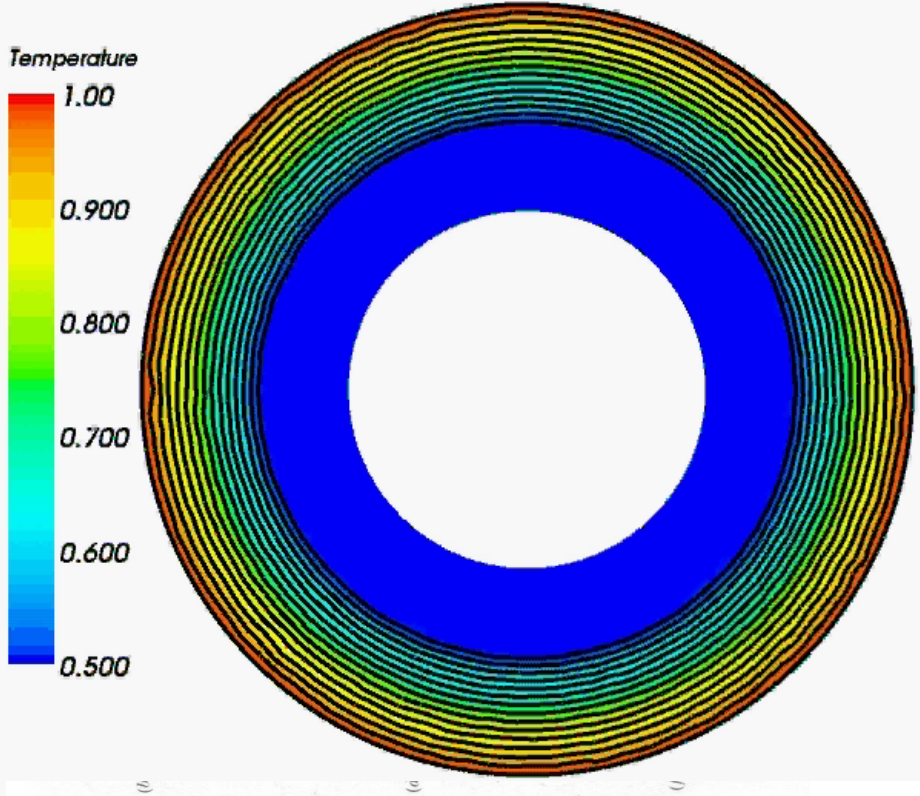


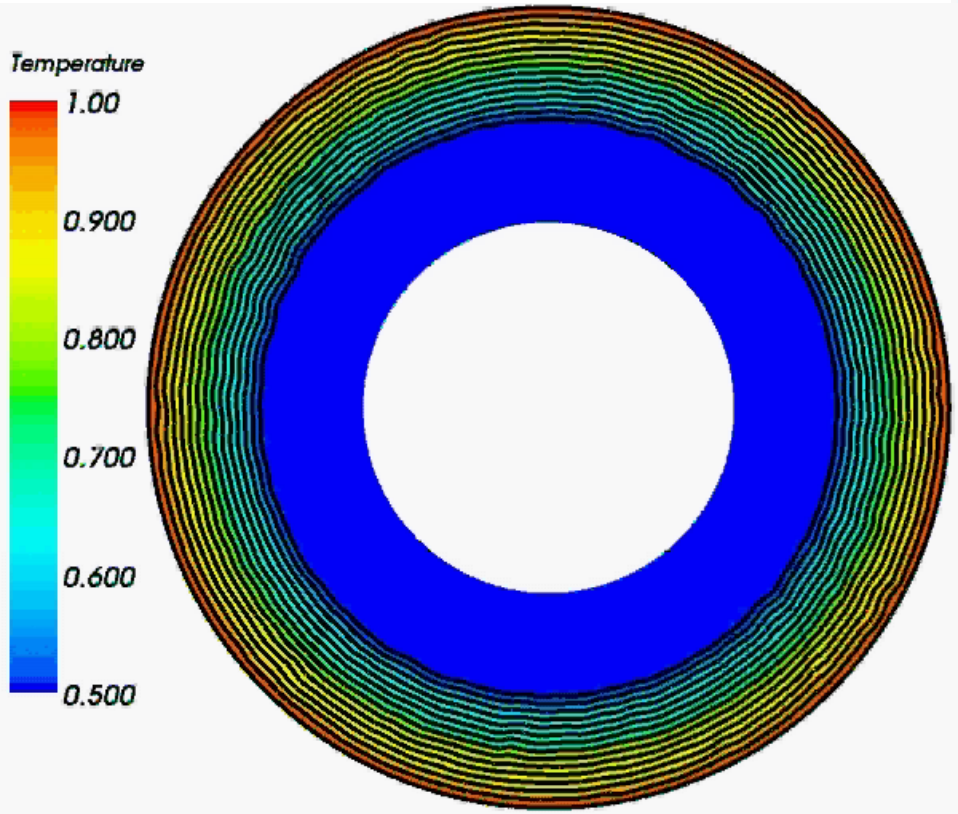


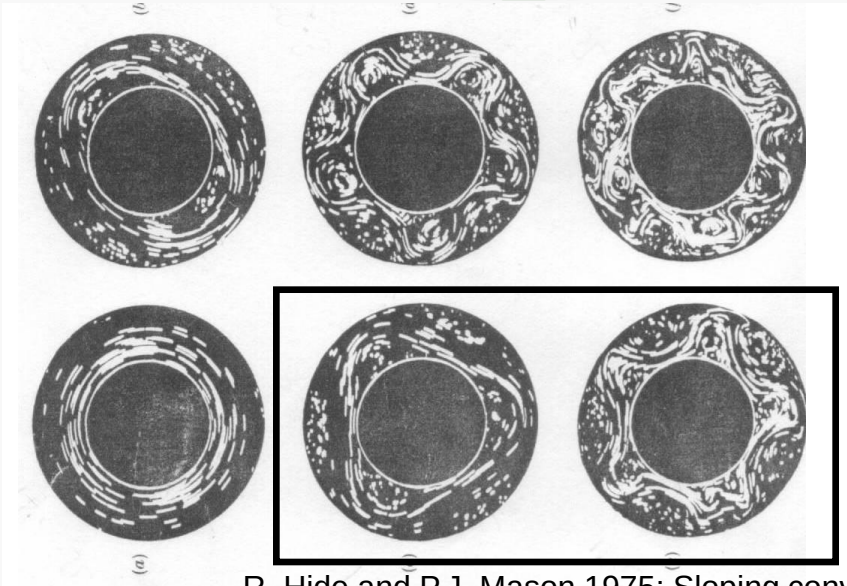


Validation against Laboratory experiments

The 3D differentially heated rotating annulus at two rotation rates. The bounds of the normalised temperature have been limited to aid visualisation at the end of the movie



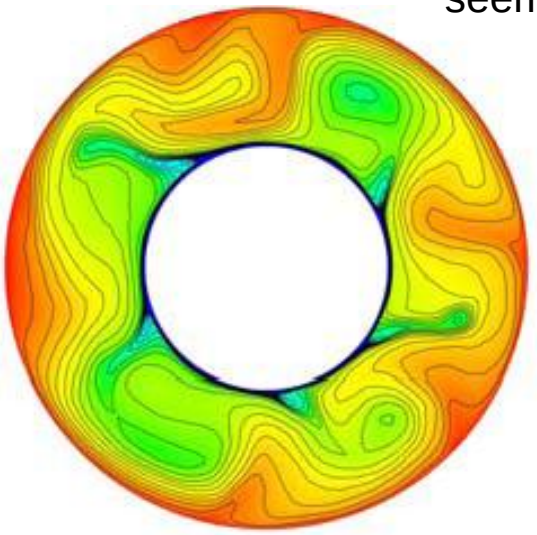


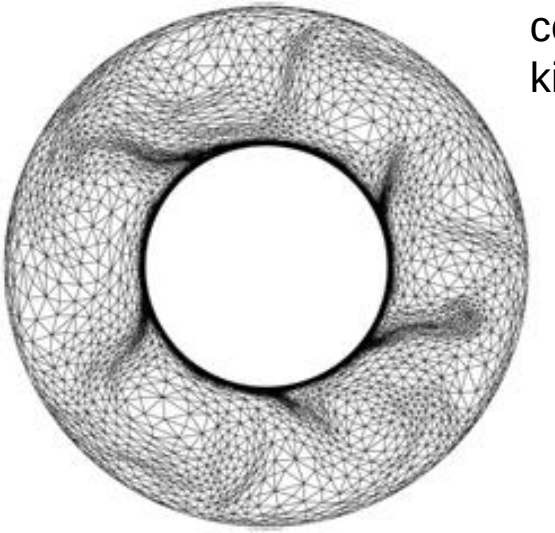


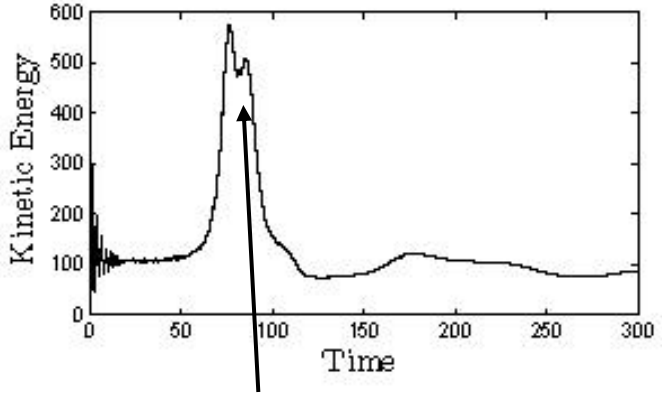
R. Hide and P.J. Mason 1975: Sloping convection in a rotating fluid. Adv. Phys. 24, 47-100

Two snapshots in time at the faster rotation rate

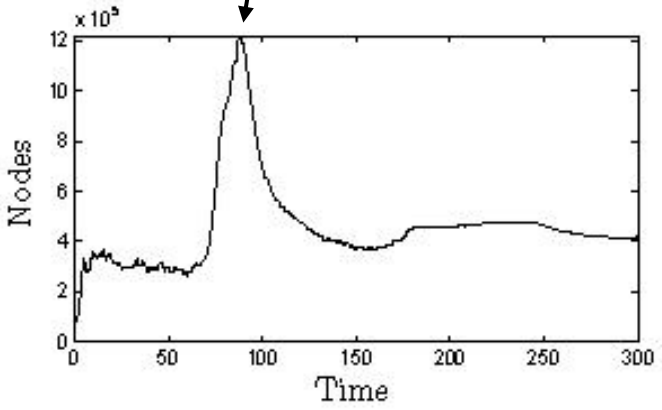
Recent quantitative comparisons with heat transport data from lab runs shows that it is hard to beat an optimised fixed mesh at lower rotation rates, but adaptivity does seem to be needed for higher rotation rates



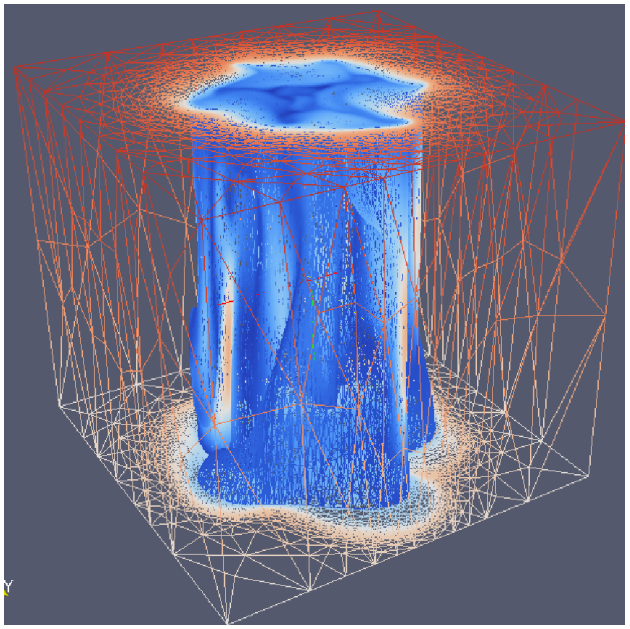


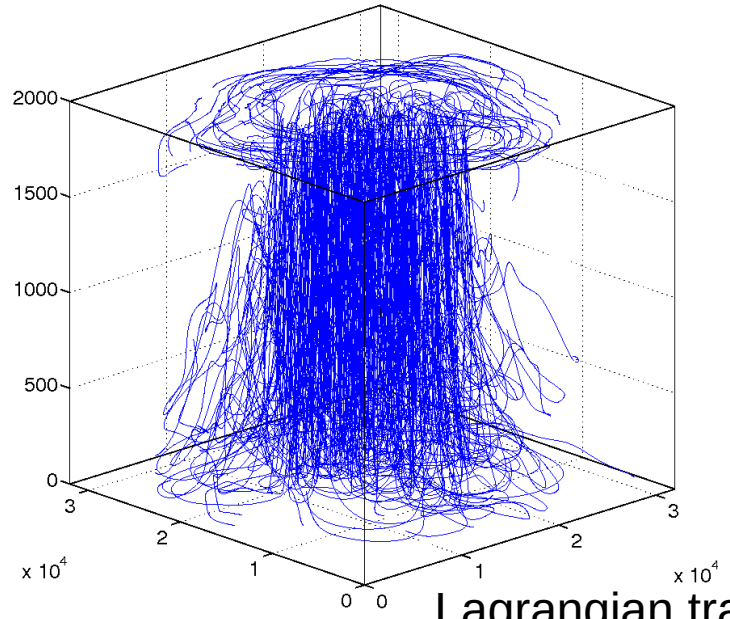


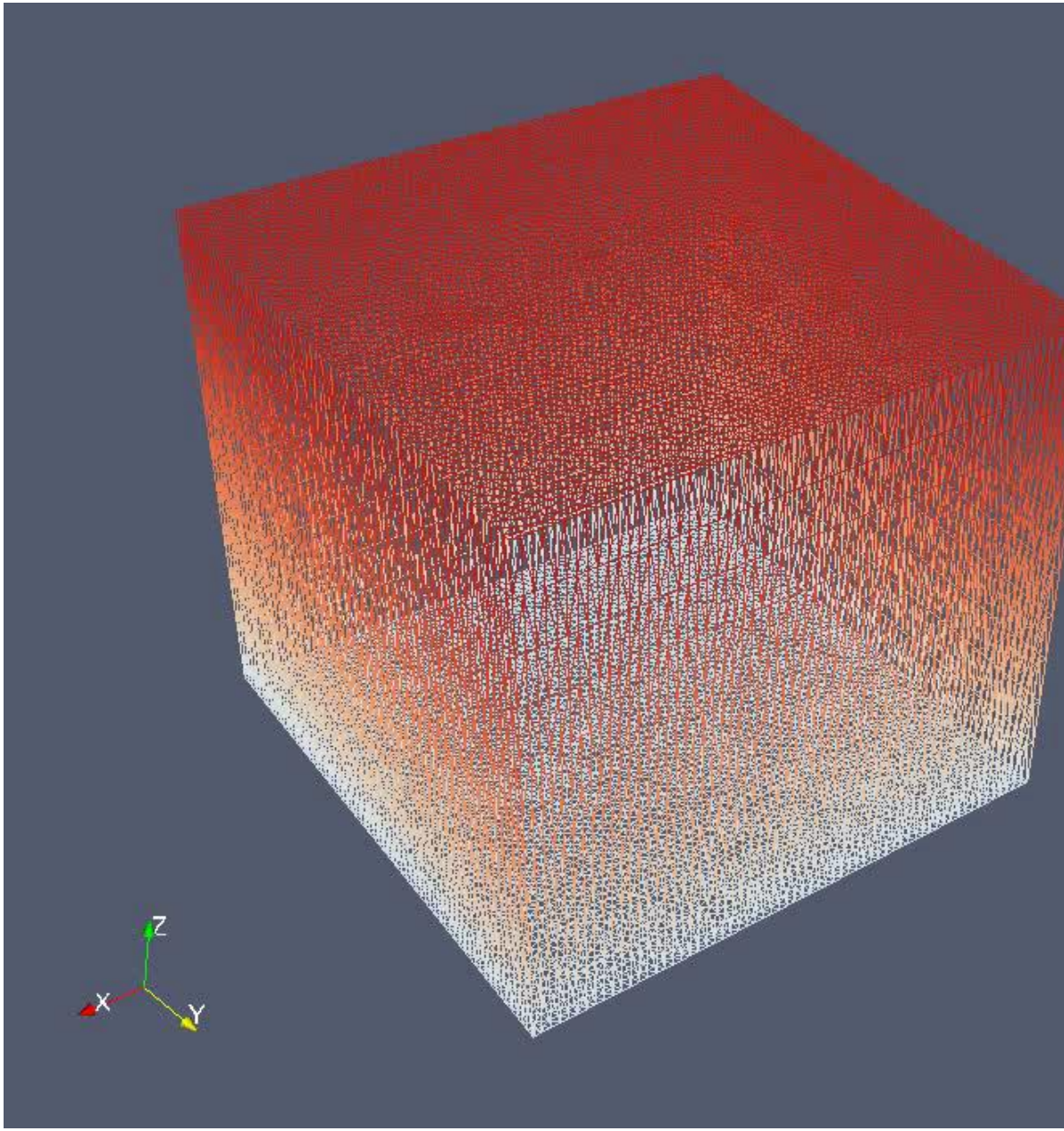
Here the maximum computational cost corresponds to the time of maximum kinetic energy



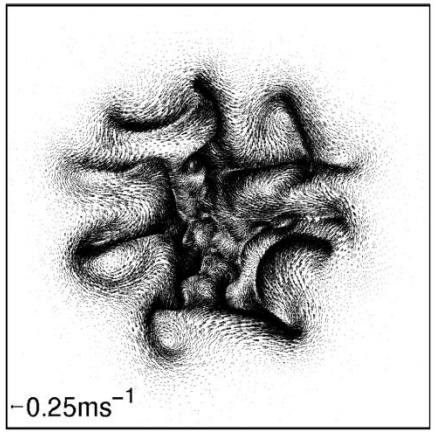
3D example with buoyancy and rotation: deep convection



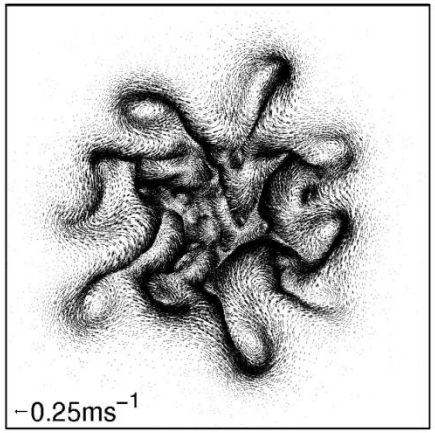




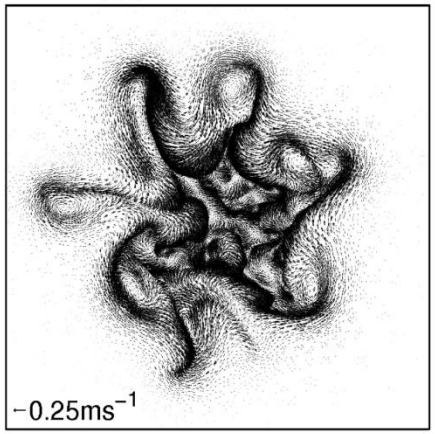
Lagrangian tracers/detectors



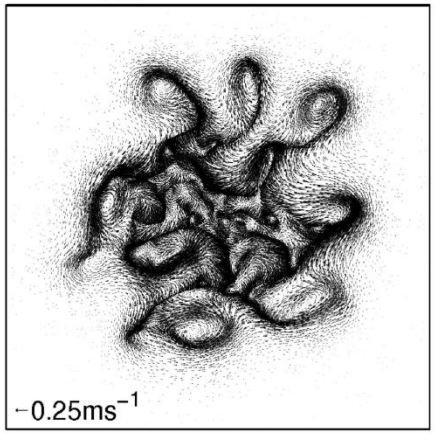
(a) $f = 0.5 \times 10^{-4}\mathrm{s}^{-1}$



(b) $f = 0.8 \times 10^{-4}\mathrm{s}^{-1}$

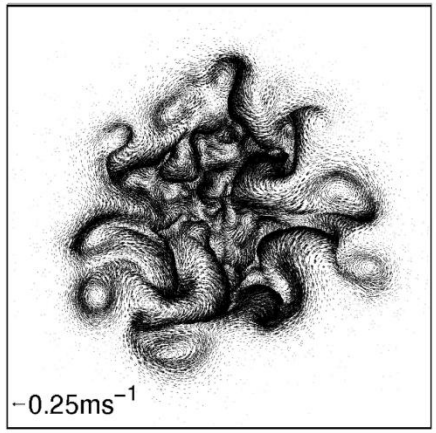


(c) $f = 1.0 \times 10^{-4}\mathrm{s}^{-1}$

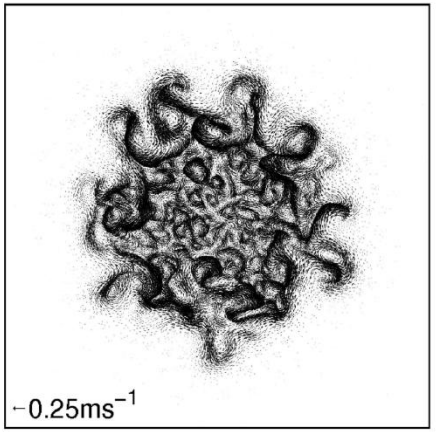


(d) $f = 2.0 \times 10^{-4}\mathrm{s}^{-1}$

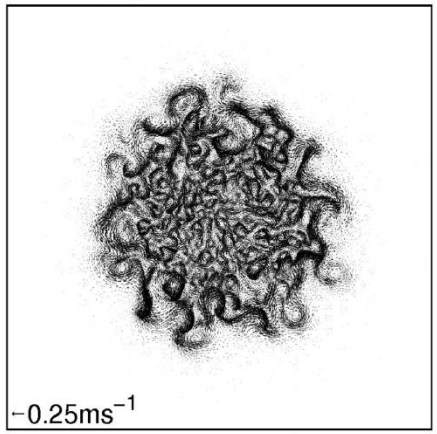
Adaptivity allows you to efficiently capture finer and finer scale motions



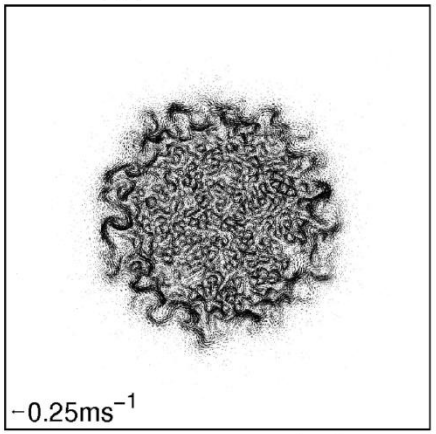
(a) $f = 3.0 \times 10^{-4}\mathrm{s}^{-1}$



(b) $f = 5.0 \times 10^{-4}\mathrm{s}^{-1}$

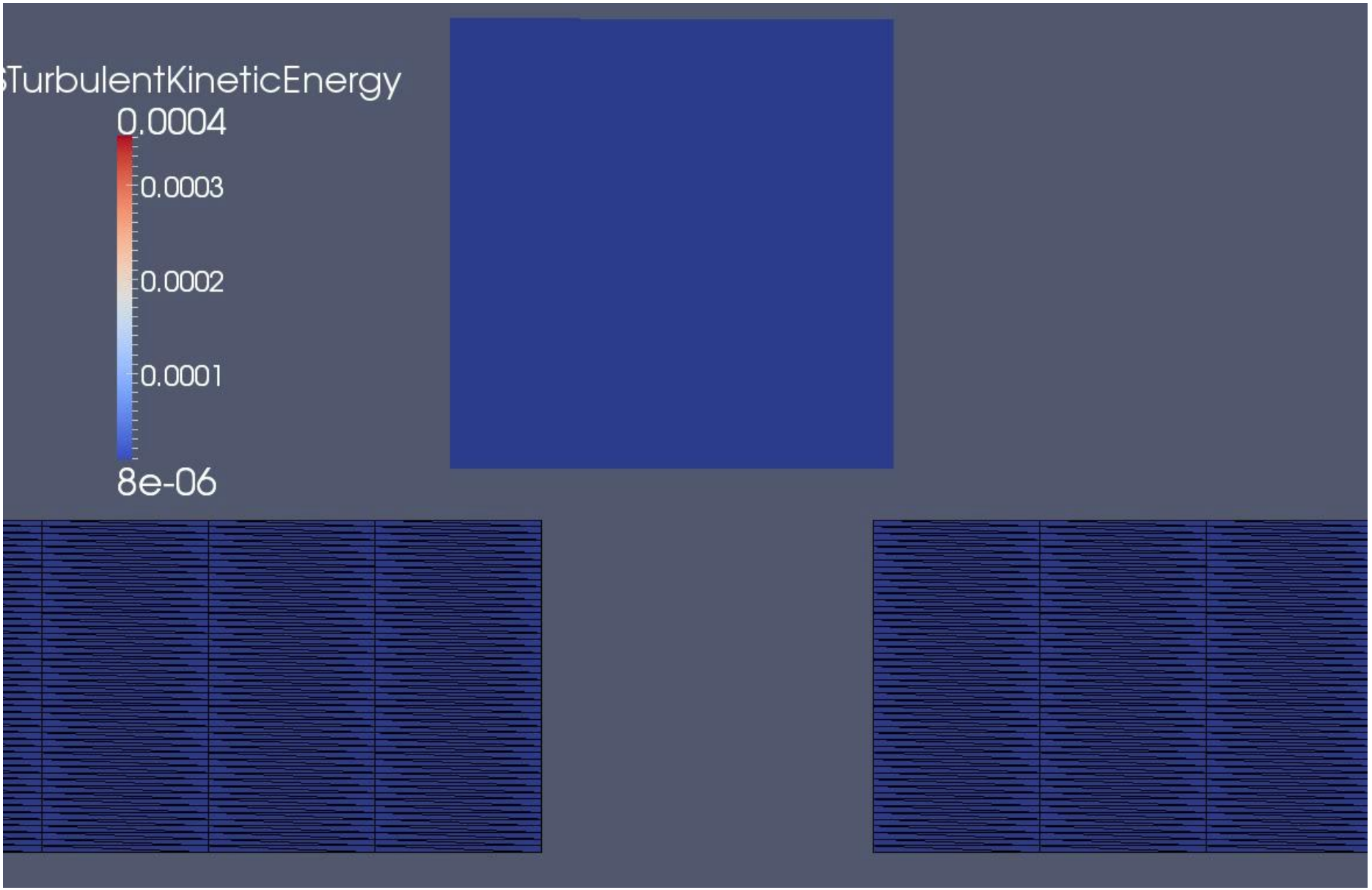


(c) $f = 7.0 \times 10^{-4}\mathrm{s}^{-1}$

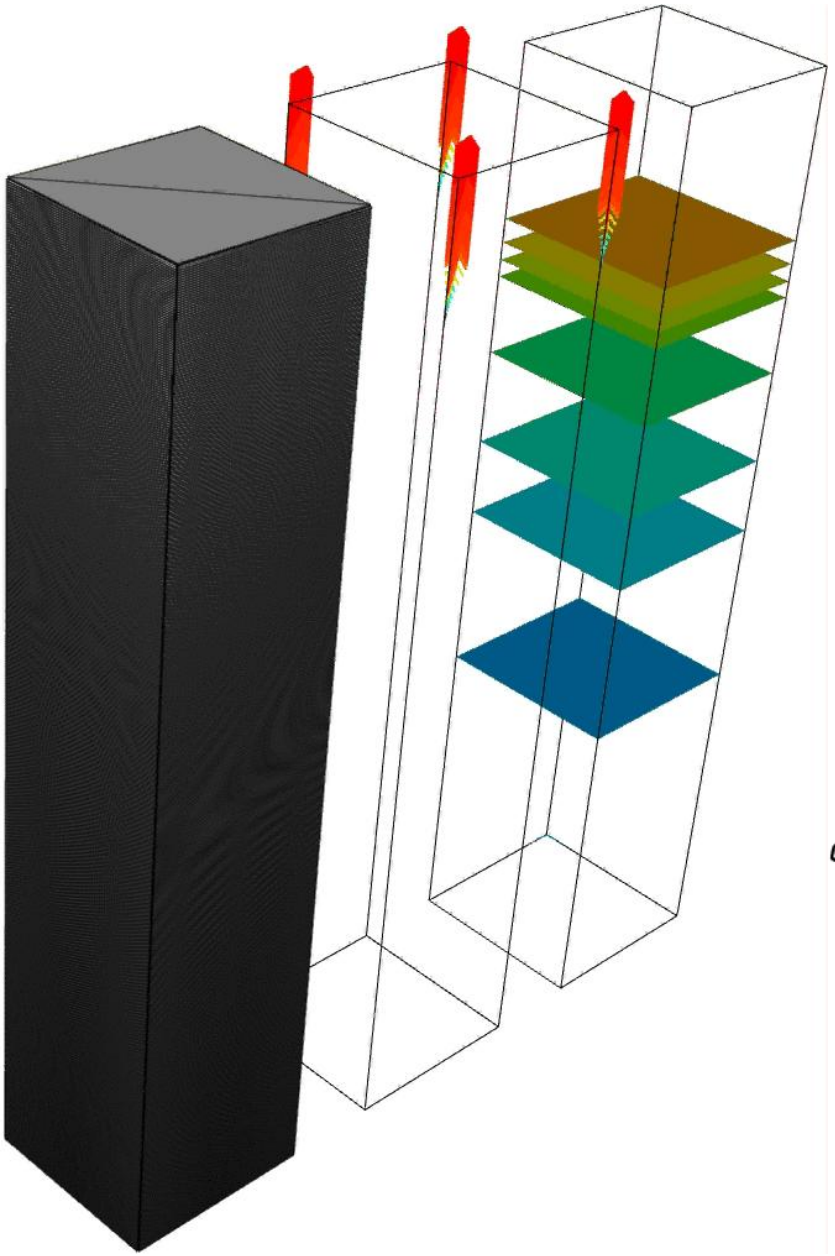


(d) $f = 10.0 \times 10^{-4}\mathrm{s}^{-1}$

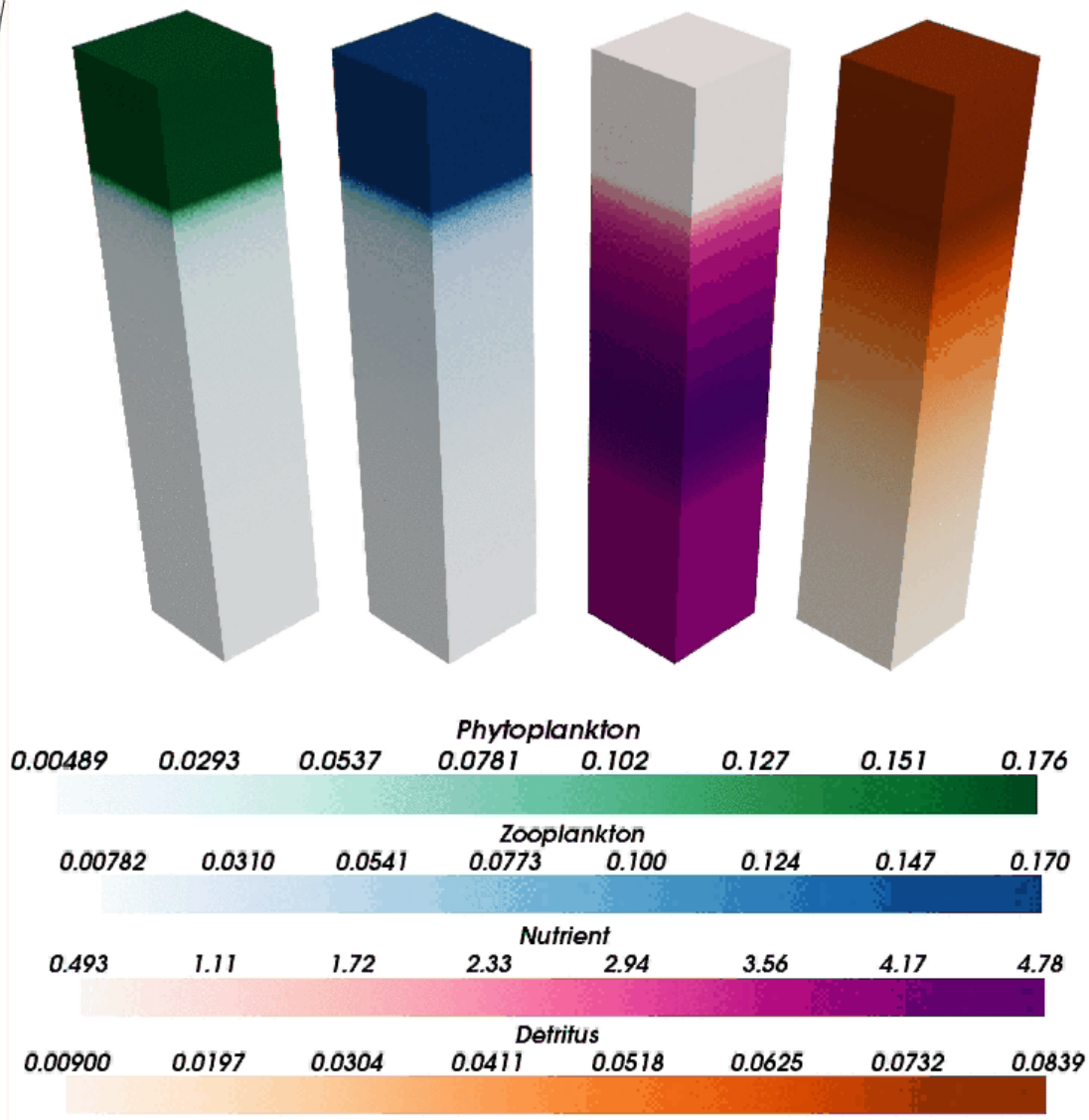
Mixed layer deepening with a RANS model - TKE and adapted mesh in a periodic domain



Physics

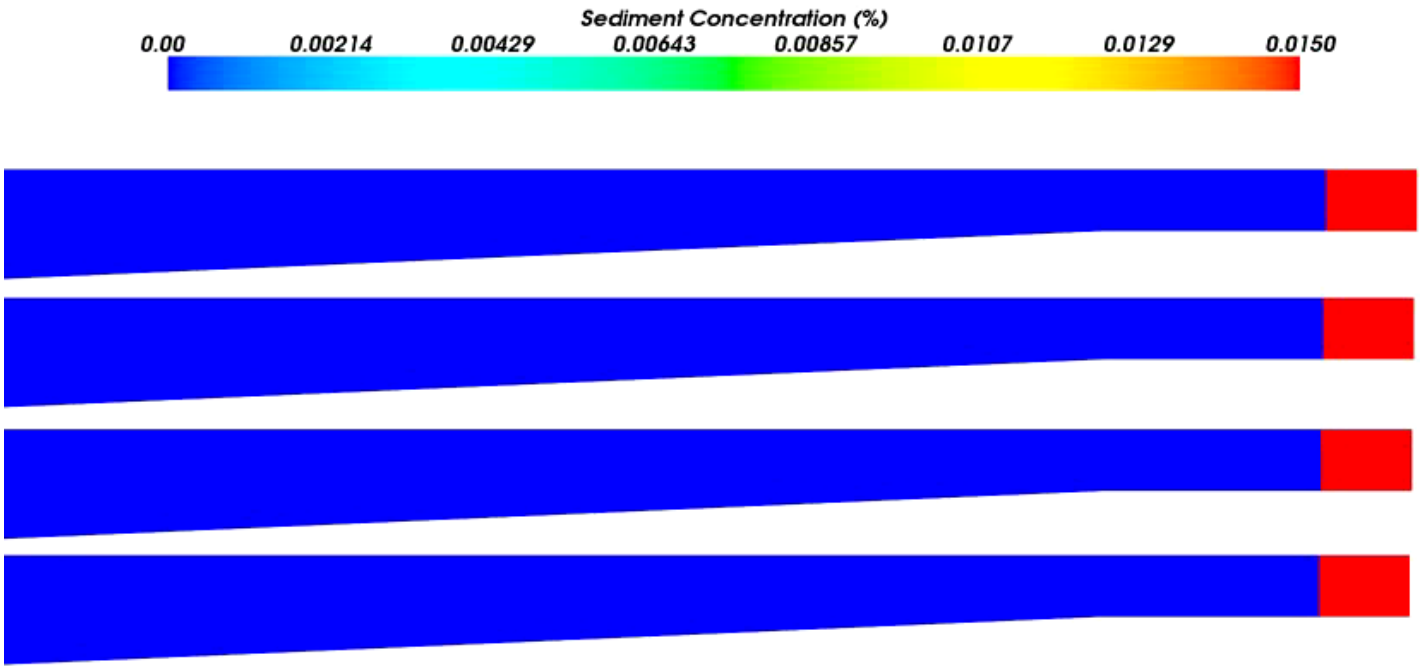


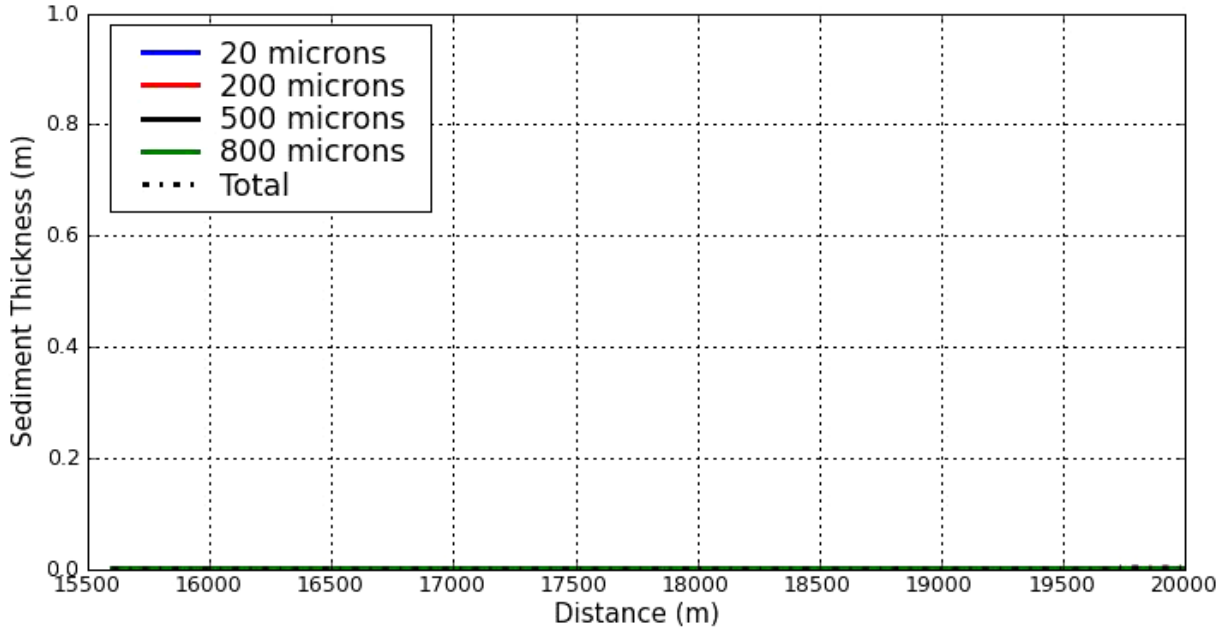
Plankton Biology



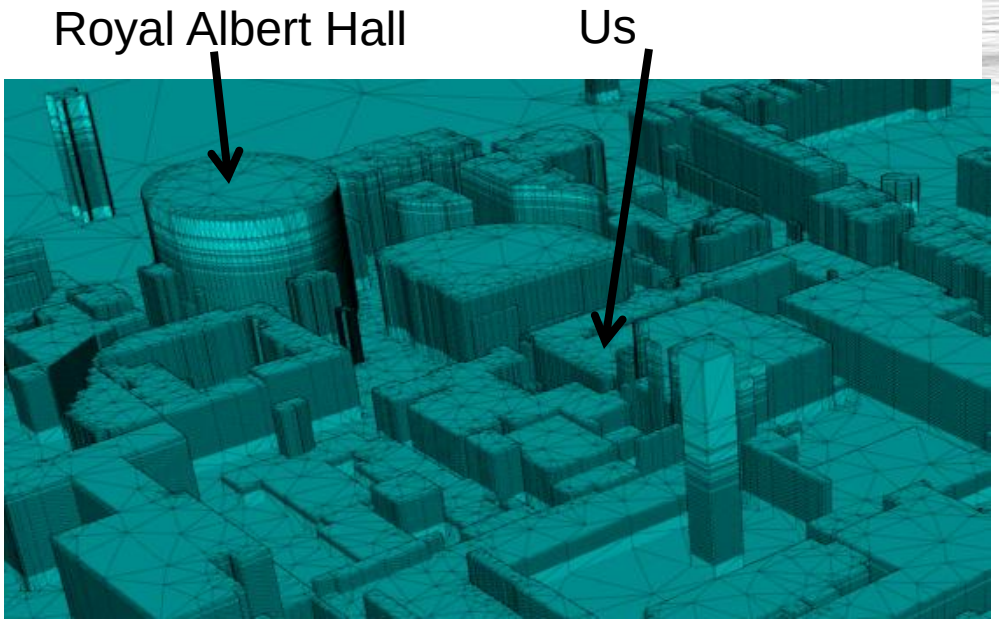
Sediment transport with 4 grain size classes

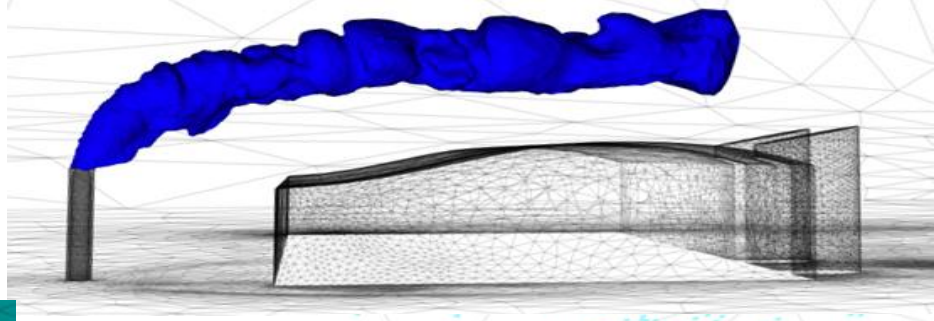
Based on our framework developed to simulate plankton biology

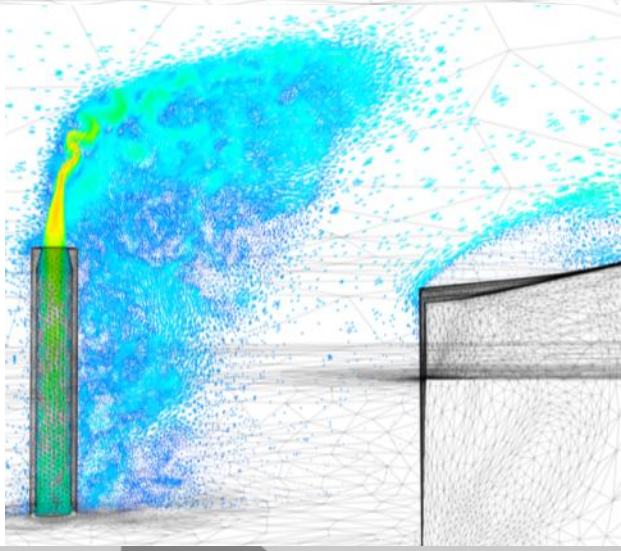


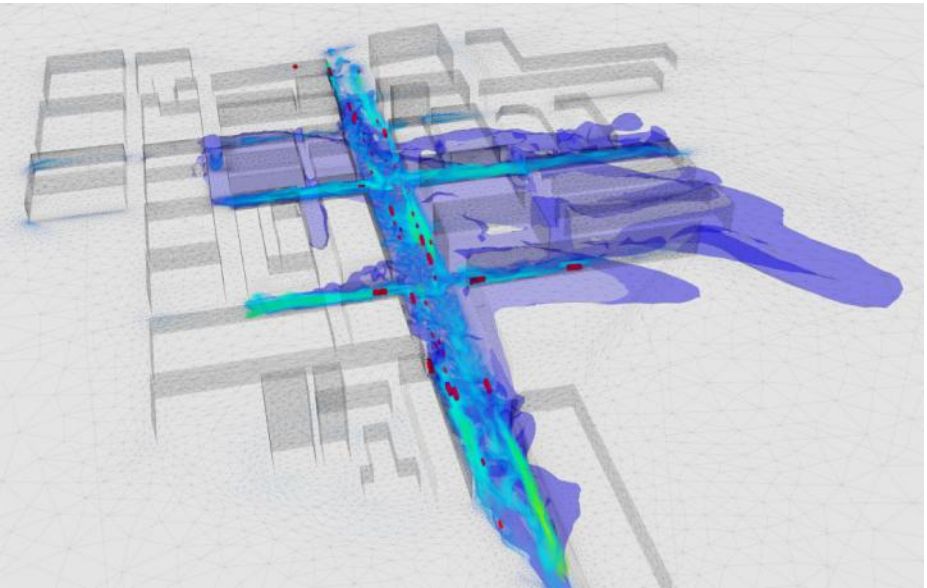


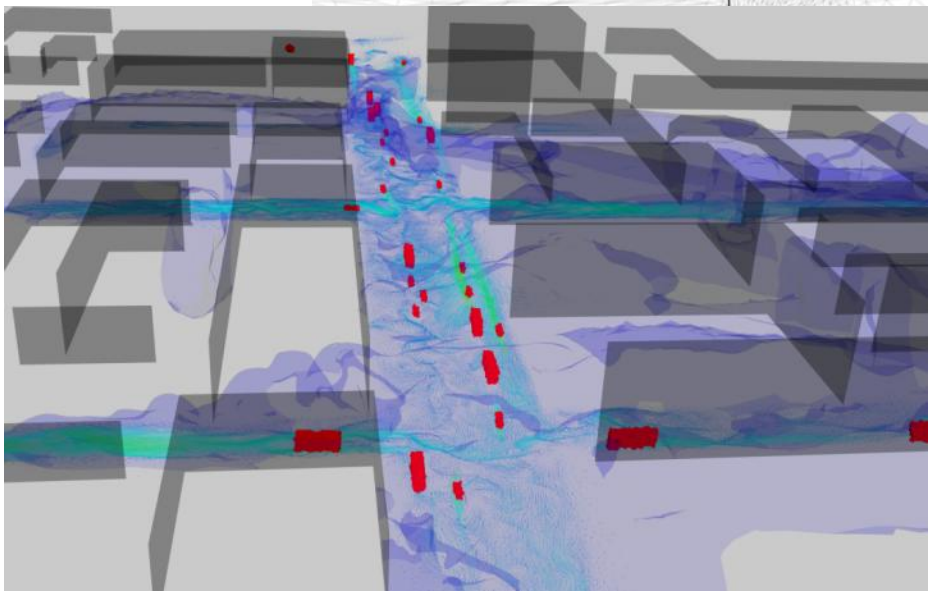
Atmospheric pollution dispersal





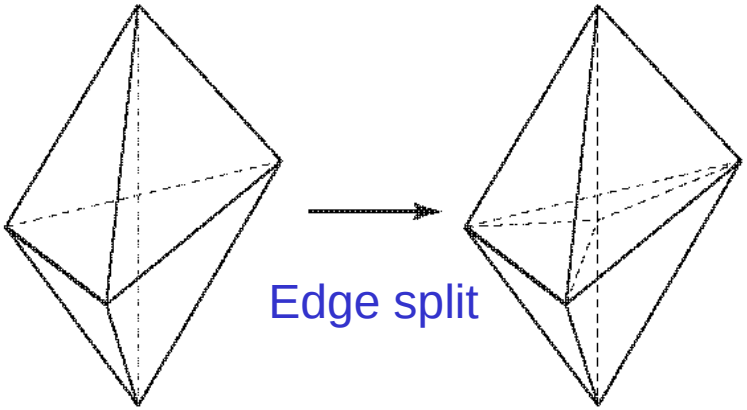




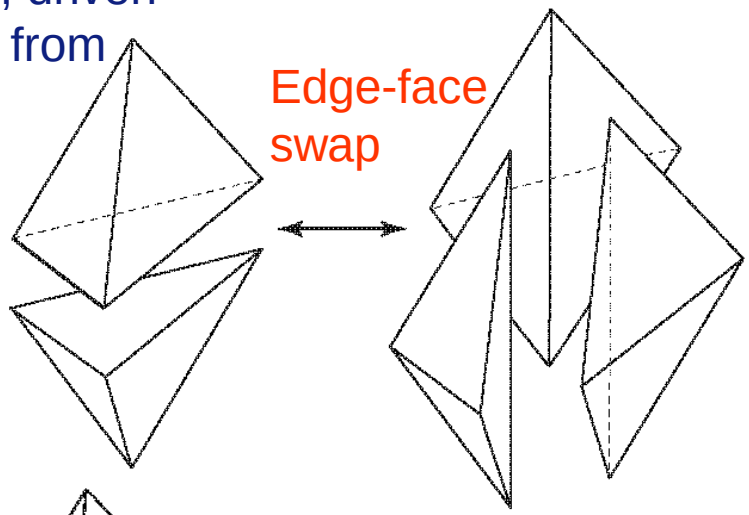


Mesh adaptivity

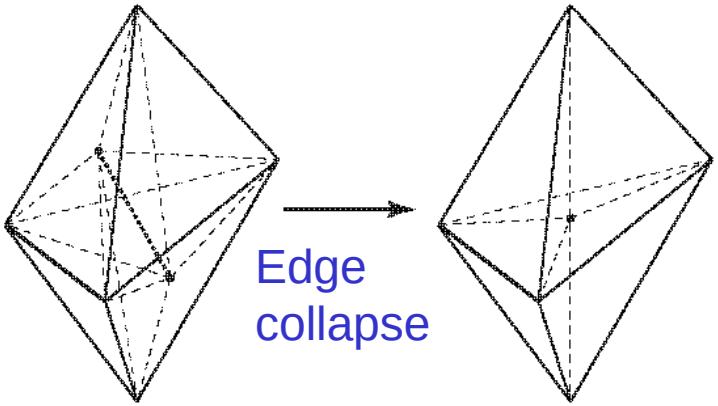
Operations used by the *mesh optimisation* algorithm, driven by an inhomogeneous & anisotropic metric obtained from error analysis (library described in Pain et al., 2001)



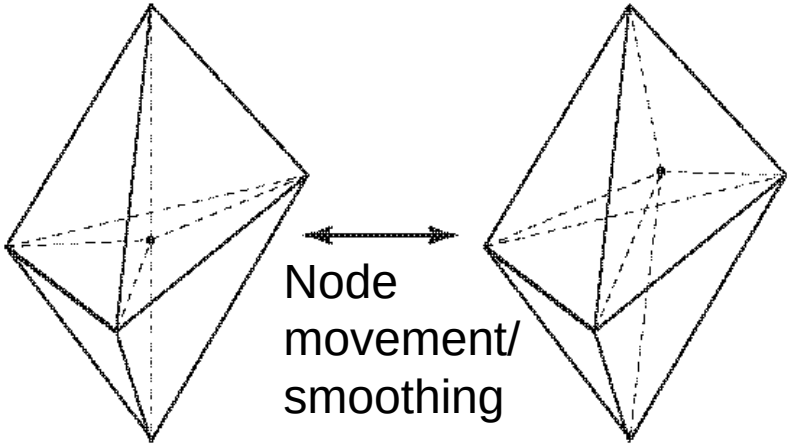
Edge split



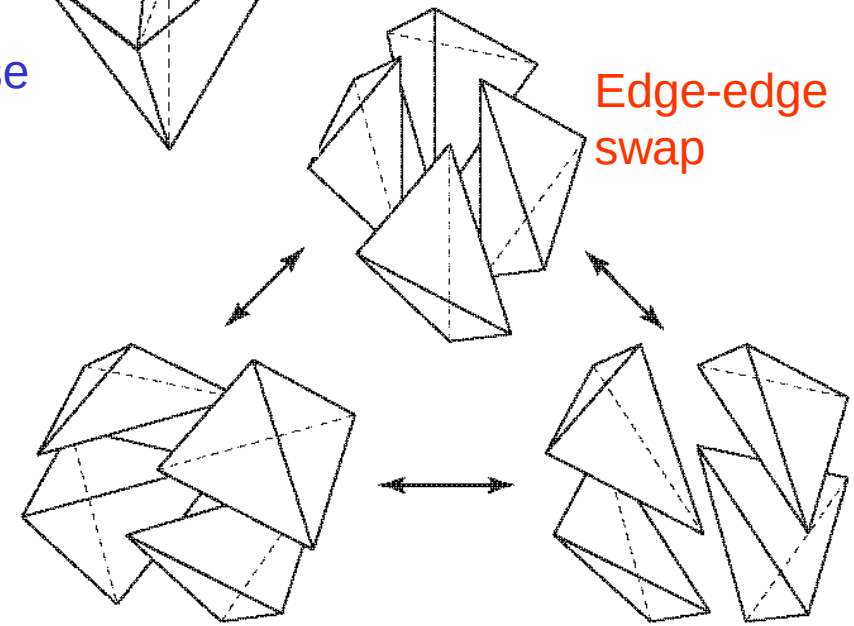
Edge-face swap



Edge collapse



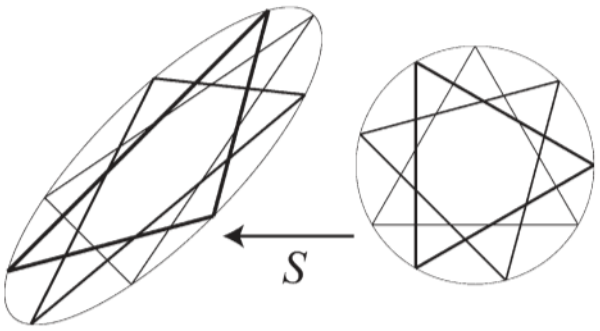
Node movement/ smoothing

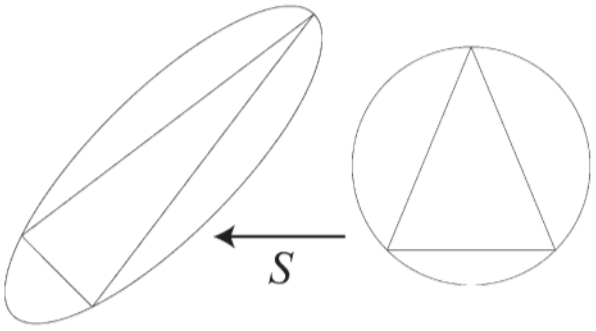


Edge-edge swap

Anisotropic mesh optimisation via a Hessian based metric

- Motivated initially by fact that discretisation errors can be linked to interpolation errors
- Linear interpolation error in 1D -> error is proportional to the mesh spacing squared multiplied by the exact solution's second derivative
- In multi-dimensions it is bounded over an element by $v^T|H|v$ where H is the Hessian matrix and v is an arbitrary vector in the element (e.g. take the edges of the element)
- We seek an adapted or optimised mesh which equidistributes this error, i.e. a mesh where all edge lengths satisfy $v^T|H|v=\varepsilon$ – a user defined tolerance
- Equivalently all elements should be equilateral and of unit size when considered in a metric (or computational) space defined by the metric $M=|H|/\varepsilon$





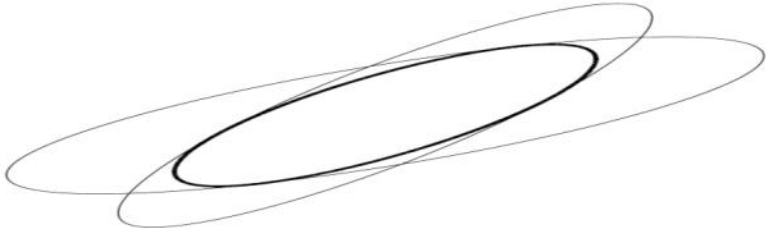
Optimal elements in physical space are those which are the realisation of equilateral elements under a coordinate transform obtained from the Hessian (directional stretching)

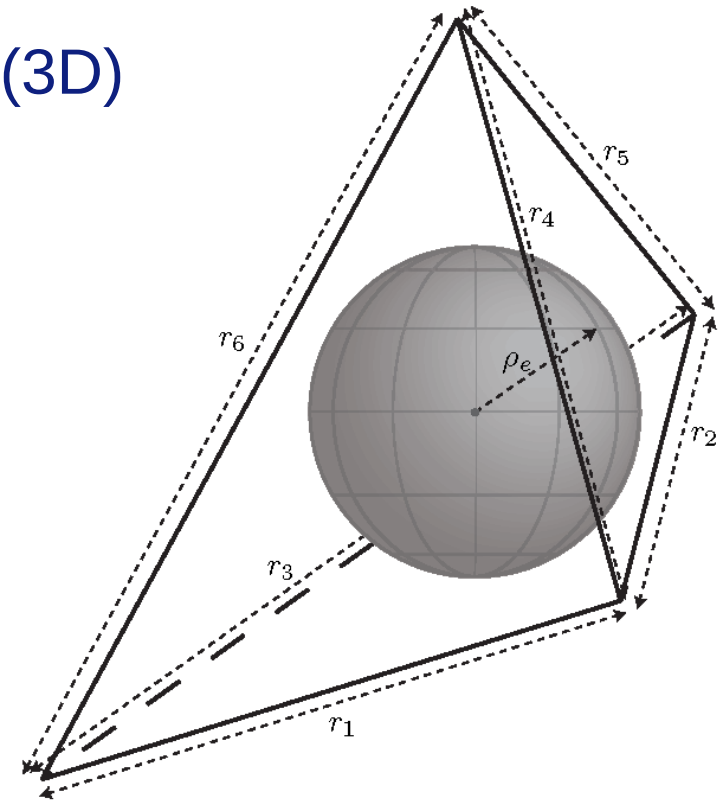
Formulating an optimisation problem (3D)

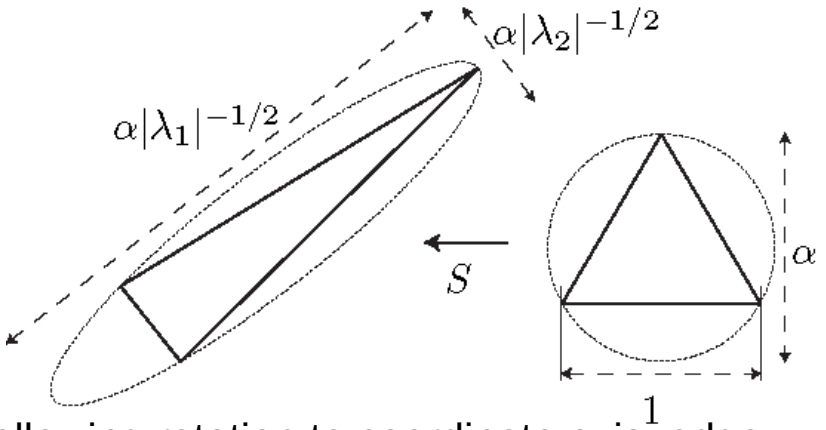
- Form an optimisation functional, in 3D we use:

$$F_e = \frac{1}{2}\sum_{l\in\mathcal{L}_e}(r_l - 1)^2 + \left(\frac{\alpha}{\rho_e} - 1\right)^2$$

- First term ensures good edge length, second term helps with element shape
- Evaluate with respect to a metric $M = \frac{|H|}{\epsilon}$ where H is the Hessian matrix, motivated by linear interpolation error
- Minimise the functional through local mesh operations
- The total metric is actually formed by combining metrics for each solution field, incorporating user-defined weightings of each, and applying constraints of max/min edge lengths, max. number of nodes and gradation to ensure smooth variation of mesh sizes





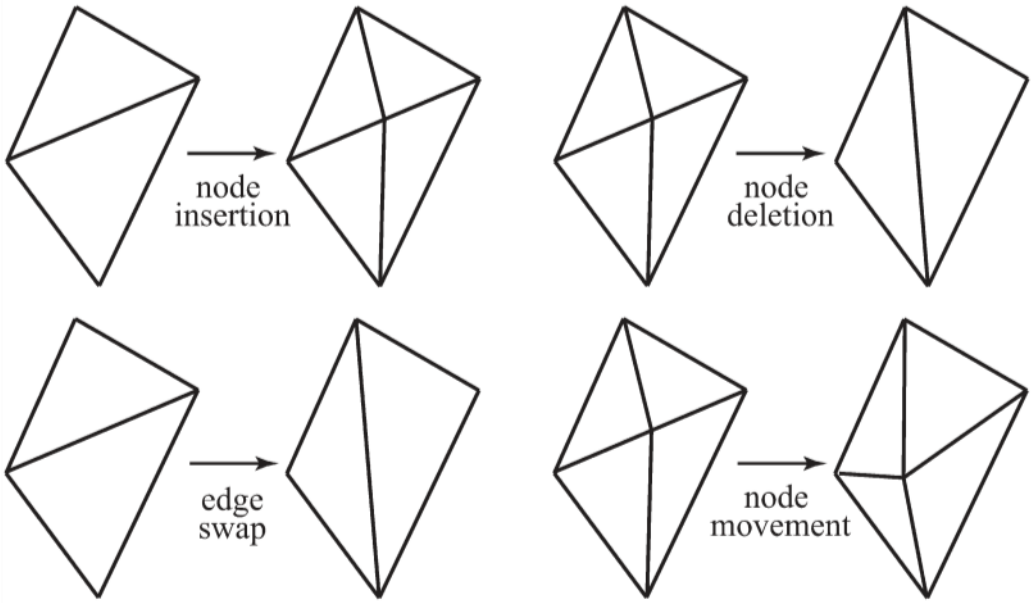


Following rotation to coordinate axis, edge lengths still satisfy:

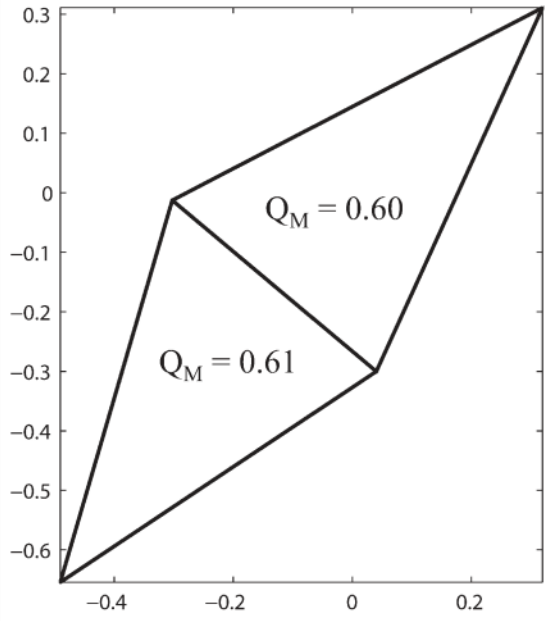
$$\Delta x^2 \frac{\partial^2 u}{\partial x^2} = \Delta y^2 \frac{\partial^2 u}{\partial y^2} = \epsilon$$

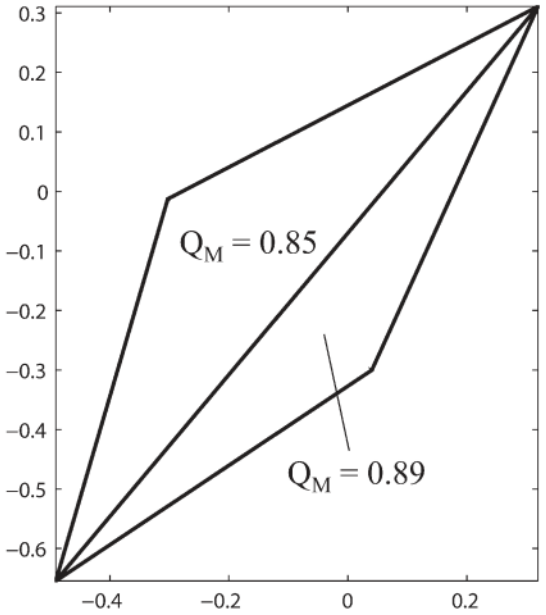
Two-dimensional mesh optimisation

For 2D applications we use the approach of Lipnikov et al. and their libmba library



E.g. An example of the effect of an edge swap on the element quality functional for a pair of elements in 2D. The 2D functional is optimal for a value of unity here





Metric

Some recent references

- Piggott MD, Gorman GJ, Pain CC, Allison PA, Candy AS, Martin BT, Wells MR, A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes, International Journal for Numerical Methods in Fluids 56, 1003 - 1015, 2008, doi:10.1002/fld.1663
- Piggott MD, Pain CC, Gorman GJ, et al, h, r, and hr adaptivity with applications in numerical ocean modelling, Ocean Modelling 10, 95 - 113, 2005, doi:10.1016/j.ocemod.2004.07.007
- Piggott MD, Farrell PE, Wilson CR, Gorman GJ, Pain CC, Anisotropic mesh adaptivity for multi-scale ocean modelling, Phil. Trans. R. Soc. A 367, no. 1907, 4591-4611, 2009. doi:10.1098/rsta.2009.0155
- Gorman GJ, Pain CC, Piggott MD, Umpleby AP, Farrell PE, Maddison JR, 2009: Interleaved parallel tetrahedral mesh optimisation and dynamic load-balancing, Adaptive Modeling and Simulation 2009, 101-104, Bouillard Ph, Diez P (eds.), CIMNE.
- Farrell PE, Piggott MD, Pain CC, Gorman GJ, Wilson CR, Conservative interpolation between unstructured meshes via supermesh construction, Comp. Meth. Appl. Mech. Eng., 198, 2632-2642, 2009. doi:10.1016/j.cma.2009.03.004
- Pain CC, Umpleby AP, de Oliveira CRE, Goddard AJH, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Computer Methods in Applied Mechanics and Engineering, 190 (29-30), 3771-3796, (2001), doi:10.1016/S0045-7825(00)00294-2
- Mitchell AJ, Allison PA, Piggott MD, Gorman GJ, Pain CC, Hampson GJ, Numerical modelling of tsunami propagation with implications for sedimentation in ancient epicontinental seas: the Lower Jurassic Laurasian Seaway, Journal of Sedimentary Geology 228, 81-97, 2010. doi:10.1016/j.sedgeo.2010.03.008
- Wells MR, Allison PA, Piggott MD, Hampson GJ, Pain CC, Gorman GJ, Tidal modelling of an ancient tide-dominated seaway, part 1: model validation and application to global mid Cretaceous (Aptian) tides, Journal of Sedimentary Research 80, 393-410, 2010. doi:10.2110/jsr.2010.044
- Farrell PE, Piggott MD, Gorman GJ, Ham DA, Wilson CR, Automated continuous verification and validation for numerical simulation, Geoscientific Model Development Discussions, 2010, Vol:3, Pages:1587-1623. http://dx.doi.org/10.5194/gmdd-3-1587-2010
- Ham DA, Farrell PE, Gorman GJ, Maddison JR, Wilson CR, Kramer SC, Shipton J, Collins GS, Cotter CJ, Piggott MD, Spud 1.0: generalising and automating the user interfaces of scientific computer models, Geosci. Model Dev., 2, 33-42, 2009. http://www.geosci-model-dev.net/2/33/2009/gmd-2-33-2009.html
- Stephan C. Kramer, Colin J. Cotter and Chris C. Pain. *Solving the Poisson equation on small aspect ratio domains using unstructured meshes*, submitted to Ocean Modelling. Preprint: http://arxiv.org/pdf/0912.1976
- Cotter CJ, Ham DA, Pain CC, Reich S, LBB stability of a mixed Galerkin finite element pair for fluid flow simulations, J COMPUT PHYS, 2009, Vol:228, Pages:336-348, http://dx.doi.org/10.1016/j.jcp.2008.09.014

Some recommended text books

- Elman HC, Silvester DJ, Wathen AJ, Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics , OUP, 2005.
- Donea J, Huerta A, Finite Element Methods for Flow Problems, Wiley, 2003
- Gresho PM, Sani RL, Incompressible Flow and the Finite Element Method (2 volumes), Wiley, 2000
- Zienkiewicz OC, Taylor RL, The Finite Element Method (3 volumes), Butterworth-Heinemann, 2005
- Frey PJ, George P-L, Mesh Generation: Application to Finite Elements, Hermes, 2000